SIMPLE SYNTHETIC DATA AS SOURCE DOMAIN FOR TRANSFER LEARNING TO REMOTE SENSING AS A TARGET DOMAIN

by

Brian L Shaw

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Geoinformatics and Geospatial Intelligence

Committee:

_____	Dr. Andreas Züfle, Thesis Chair
_____	Dr. Arie Croitoru, Committee Member
_____	Dr. Olga Gkountouna, Committee Member
_____	Dr. Dieter Pfoser, Department Chairperson
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
_____	Dr. Fernando R. Miralles-Wilhelm Dean, College of Science
Date: _____	Fall Semester 2022 George Mason University Fairfax, VA

Simple Synthetic Data as Source Domain for Transfer Learning to Remote Sensing as a Target Domain

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Brian L Shaw

Director: Andreas Züfle, Associate Professor
Department of Geography and Geoinformation Science

Fall Semester 2023
George Mason University
Fairfax, VA

DEDICATION

This is dedicated to the person for whom I am most thankful. That is, I am genuinely grateful to my remarkably gracious wife, Ellie, perhaps the most patient person in the world. Yep. I just checked my notes, and she is.

ACKNOWLEDGEMENTS

I would like to thank Dr. Andreas Züfle, Dr. Arie Croitoru, and Dr. Olga Gkountouna who provided valued perspective, encouraging guidance, and savvy insight.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

Arithmetic Logic Units ..... ALU
Concept Activation Vector ..... CAV
Compressed County Mosaics ..... CCM
Generative Adversarial Networks ..... GAN
Geospatial Data Gateway ..... GDG
Ground Sample Distance ..... GSD
Hue, Saturation, Brightness ..... HSB
ImageNet Large Scale Visual Recognition Challenge ..... ILSVRC
National Agriculture Imagery Program ..... NAIP
Rectified Linear Unit ..... ReLU
Red, Green, Blue ..... RGB
structured domain randomization ..... SDR
Testing with Concept Activation Vectors ..... TCAV
Visual Geometry Group ..... VGG

ABSTRACT

SIMPLE SYNTHETIC DATA AS SOURCE DOMAIN FOR TRANSFER LEARNING TO REMOTE SENSING AS A TARGET DOMAIN

Brian L Shaw, M.S.

George Mason University, 2023

Thesis Director: Dr. Andreas Züfle

Deep Learning continues to grow as a prevalent toolset among multiple disciplines, including Remote Sensing and image analysis. Correspondingly, to more easily apply the deep neural networks to different subject matter domains, Transfer Learning, from natural image datasets, including ImageNet, has become a de-facto method for many Deep Learning applications, including Remote Sensing. However, such an approach may have limitations related to the differences on the characteristics of natural photographic image datasets and the characteristics of Remote Sensing. This study aims to determine if a fairly arbitrary, easily produced set of synthetic datasets can be iteratively developed and used for Transfer Learning for a typical Deep Learning task. We found this is readily and surprisingly feasible.

INTRODUCTION

Transfer learning from natural image datasets, particularly ImageNet, using standard deep networks and corresponding pre-trained weights, has become a de-facto method for many deep learning applications, including Remote Sensing.

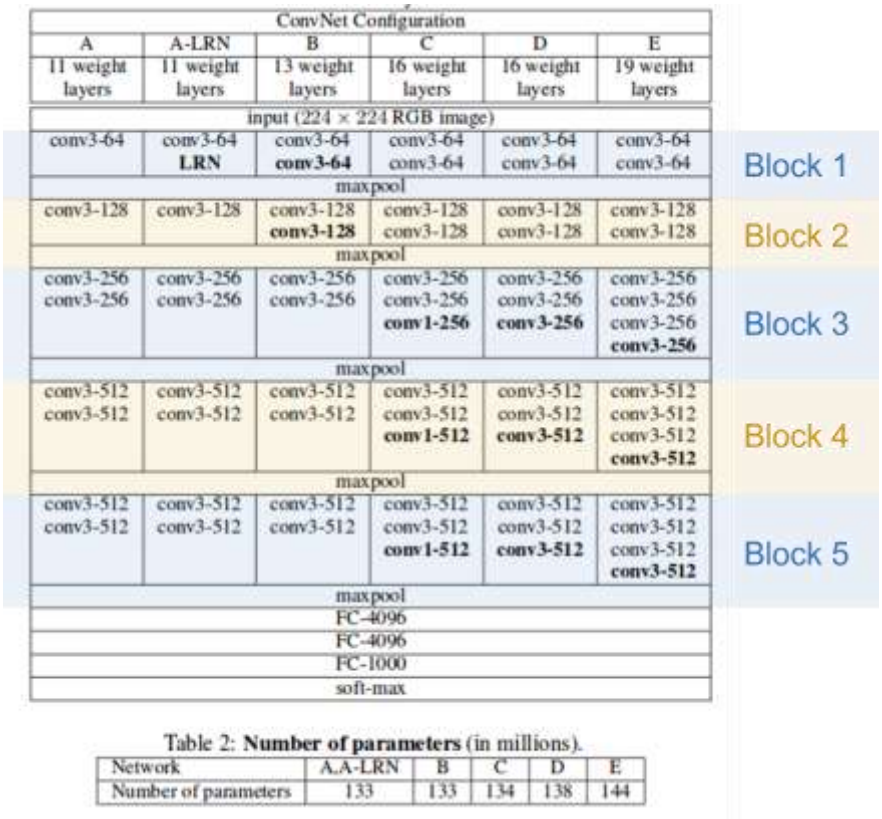
However, such an approach may have several limitations related to the differences on the characteristics of natural photographic image datasets and Remote Sensing. For example, there may be differences in data sizes, features, and task specifications between the source domain data sources and the requirements of the target domain, Remote Sensing tasks. Some of the typical solutions are to curate and develop annotated datasets such as BigEarthNet, EuroSAT, and UC Merced. Although these have been very successful, these datasets may not anticipate future requirements of downstream tasks.

To address that, this study aims to determine if a fairly arbitrary, easily produced, synthetic datasets can be developed as the source domain for Transfer Learning for a typical deep learning task. If so, then it demonstrates that researchers are not dependent upon secondary datasets. Instead, researchers can use their expert knowledge of their domains to develop their own source dataset for their requirements.

Discussion of VGG16 Design

Although CNN design is beyond the scope of this thesis, it is still informative to provide an overview of the VGG16 Convolutional Neural Network (CNN) as the design

impacts the images / tensors as they traverse through the network and can impact the design of synthetic data. VGG16 was designed by Simonyan and Zisserman of the Visual Geometry Group (VGG) from the University of Oxford to compete in an image classification competition, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ,in 2014. Although only a runner-up in ILSVRC-2014, its simplicity and modularity made a popular image classification neural network. **Figure 1** is an illustration from the authors' paper, describing the variants of their designs. The network described in Column D became known as VGG16.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Block 1
Block 2
Block 3
Block 4
Block 5

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figure 1 Architectures of VGG Configurations from Simonyan, et al. [1]

Another graphic showing a more intuitively accessible view of VGG16 is shown in **Figure 3** The '16' refers to the number of layers of trainable weights and biases, which includes the thirteen (13) convolutional layers and the three (3) fully connected layers. The five (5) pooling layers derive their values from the prior convolutional layers. These layers and other attributes will be described in the next section.

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

Convolution Layers – Different than many predecessor neural networks[2], the VGG team selected to use relatively small 3x3 kernels in each convolution layer, and stack multiple convolutions in sequence, before pooling the result. In each convolution layer, the 3x3 kernels are applied across the input tensor. Notice the value of the output 'pixel' represents values from nine pixels in a 3x3 area in the original image. Note this, to consider a 3x3 region on the input image, a 3x3 kernel needs to scan an area that measures 5x5. So, a 3x3 convolutional output represents a 5x5 part of the input image.

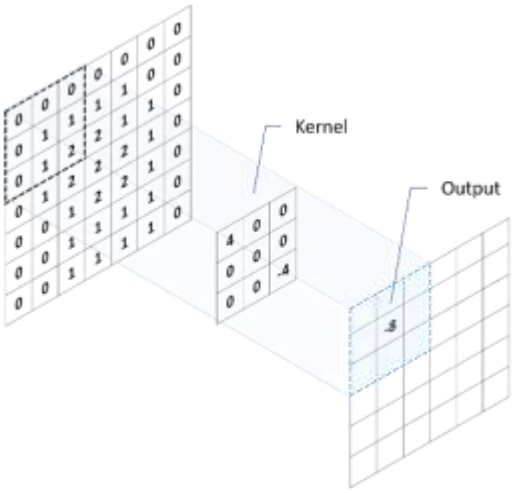


Figure 2 Conceptual graphic depicting a convolution process.

[2] Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

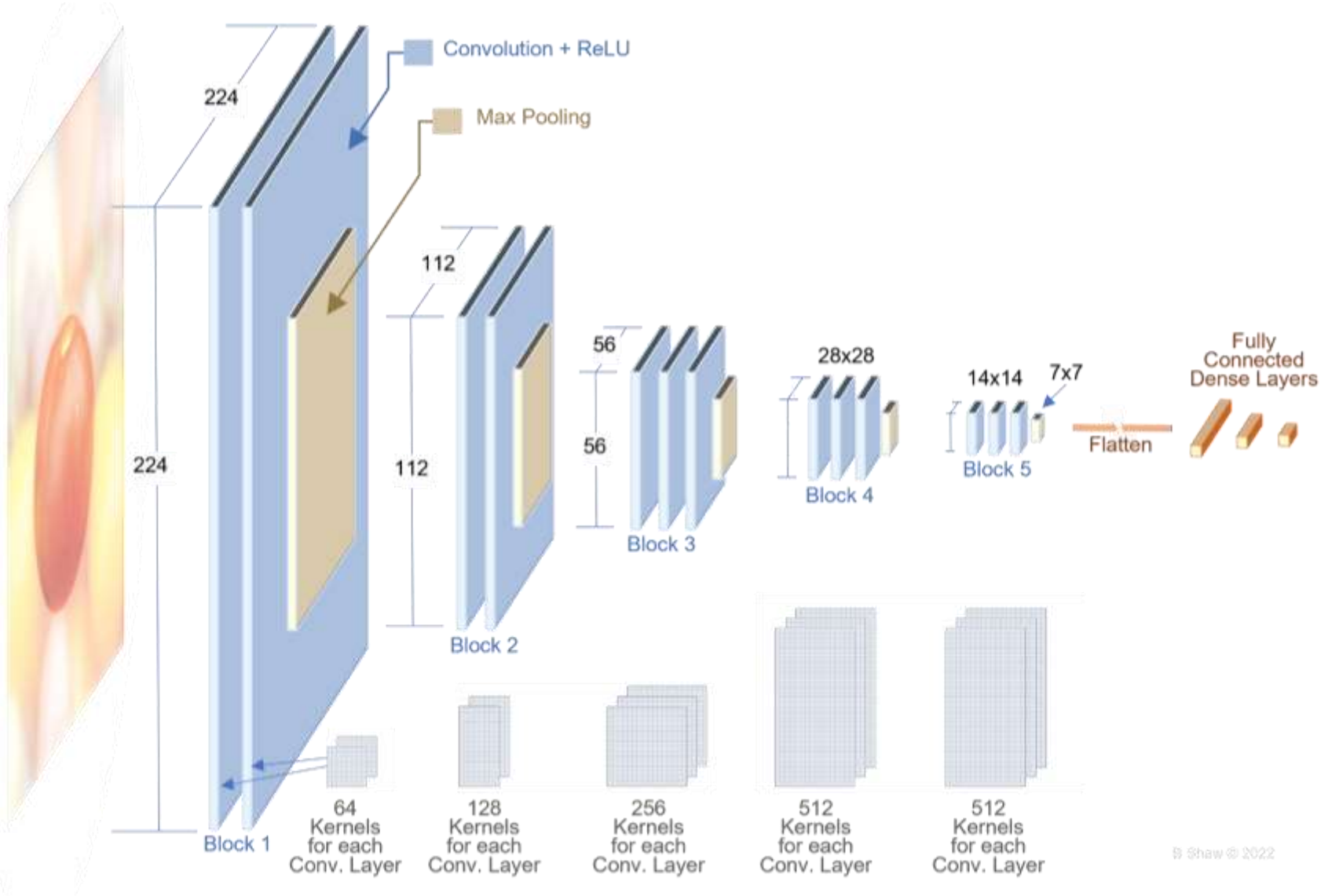


Figure 3 Illustration of VGG16 showing the convolutional and max pooling layers.

Kernel Weights and Bias – In the case of VGG16, in the very first convolution layer, the input is a three-dimensional tensor, with dimensions of 224x224x3. The three dimensions of height, width, and color channels, respectively. That is, the input tensor is a color image, with Red, Green, Blue as the three-color channels. Each 3x3 kernel is applied at each color channel at each pixel within the image. In our implementation, a single pixel of padding is added to the edges to allow of calculation of all the image area.

An element-wise product between each element of the kernel and the input tensor (i.e. color image for the first convolution layer) is calculated at each location of the tensor (i.e. pixel) and summed to obtain the output value in the corresponding position of the output tensor. This output tensor is called a feature map, and it could be considered a kind of image. In the case of VGG16, convolution layer 2 of block 1 will repeat the process with the resulting output tensor from the convolution layer 1. That is, for VGG16, all 64 kernels of convolution one (1) raster of over the image based on a defined stride, and then all 64 kernels of convolution layer 2 raster over the output tensor / feature map from convolution layer 1.

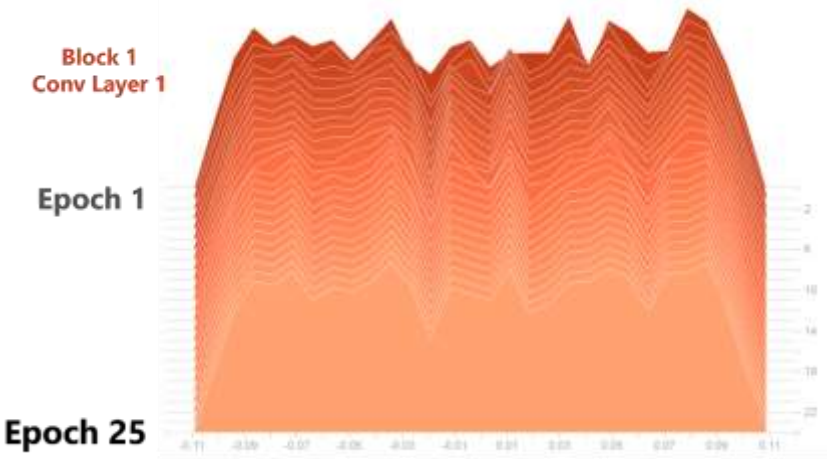


Figure 4 Histogram of kernel weights for Block1, Convolution layer 1.

In our implementation of VGG16, the values in the kernels, called Weights, start from a Glorot Uniform distribution of values. Initializing with a Glorot distribution [3] allows for a quicker convergence compared to initializing the values with a truly normal, random distribution. As the network trains, the values in the kernels are updated to lower the loss during backpropagation, using Adam as the gradient descent optimizer, in our implementation. **Figure 4** shows the histogram of weights for all the kernels in the first convolutional layer of Block 1, as it progressed during 25 epochs of training. Notice the relatively wide distribution of values, even from the first training epoch.

Likewise, **Figure 5** shows the histogram of weights for all the kernels in the first convolutional layer of Block 5, as it progressed during 25 epochs of training.

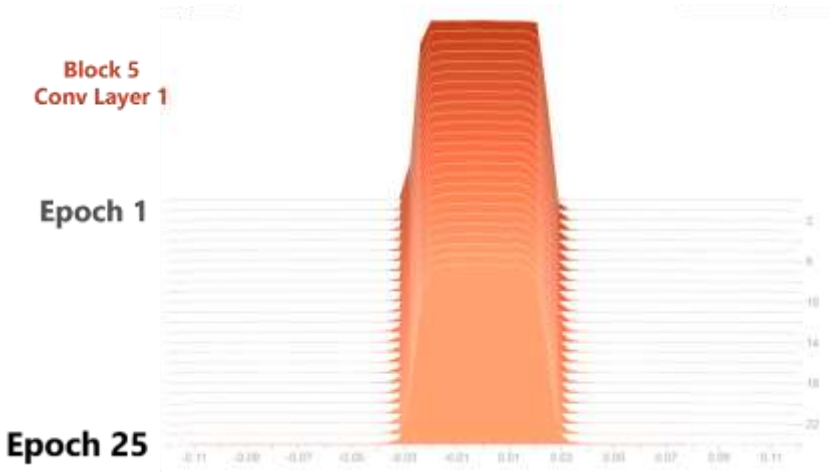


Figure 5 Histogram of kernel weights for Block5, Convolution layer 1.

Notice the breadth of the histogram values in Block 1, Convolutional Layer 1, from -0.11 to +0.11, compared to Block 5, Convolutional Layer 1, where the breadth is extends from -0.035 to +0.035. This is indicative of the typical decreasing variance around zero for

[3] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249-256. JMLR Workshop and Conference Proceedings, 2010.

each successive convolutional layer. That is, as shown here, the kernels of block 1, the initial convolution layers closest to the image input show the greater spread of weight values. The final convolutional layers in block 5, shown here, reveal the histograms of kernel distributions closer to Glorot Uniform initialization values. For our simple 3-class training, it's an indication that most of the learning is done in the first few layers and could even indicate a simpler, shallower model would be sufficient.

Why stack the layers in VGG16? Remember when we mentioned that a 3x3 convolutional output represents a 5x5 part of the input image. That 5x5 part of the input image (tensor) is called a 'receptive field' and in VGG16, the stacked layers allow a wider "receptive field". That is, a 3x3 area in the first convolution layer represents a 5x5 area from the layer below. If we were to stack another layer above, with a 3x3 kernel, that 3x3 kernel will represent a 7x7 area from two layers below, and so forth.

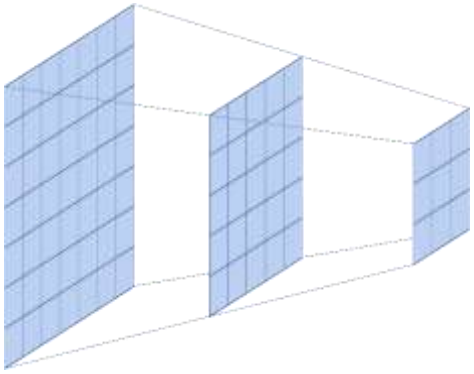


Figure 6 The benefits of a deep block of convolutional layers

Fully Connected, Dense Layers - A fully connected, dense layer is a layer in which every input neuron is connected to every output neuron in the next layer. In our VGG16 model, the resulting feature maps from the final Block 5 pooling layer is flattened then fed into the fully connected layers. That is, it is transformed into a one-dimensional array of numbers, like a spaghetti data model and then fed into three fully connected, dense layers, sequentially. The original VGG16 model used 4096 kernels for

each of the first two dense layers, and 1000 kernels for the last dense layer, to match the 1000 classes they were handling. We have only three (3) classes, so our VGG16 implementation uses 64 kernels for the first dense layer and 32 kernels for the second dense layer. For the final layer, our implementation uses 3 kernels to match the number of classes. The first two layers use the aforementioned Rectified Linear Unit (ReLU) activation, the final fully connected, dense layer, uses a linear activation.

Activations - The output of every convolution layer passes through a non-linear activation function. In our case, for VGG16, we used the ReLU after all the convolution layers. The ReLU piecewise linear function acts as a gate that only allows positive values to pass. That is, it will pass any input value if it is positive, otherwise, it will output zero. Other activation functions frequently used are SoftMax, Tanh and sigmoid for various uses (e.g., Binary Classification, Multiclass Classification), for various network types (e.g., Multilayer Perception, Recurrent Neural Networks), or to handle negative values (Leaky ReLU, Exponential LU).

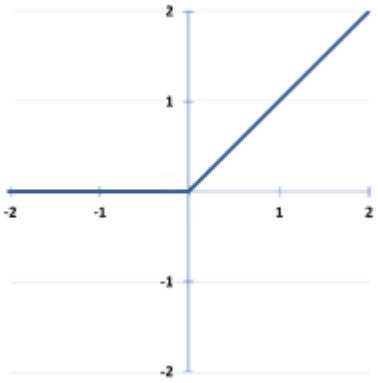


Figure 7 ReLU function allows only positive values to pass.

Logits – This is the vector of raw, non-normalized predictions that a classification model generates. Typically, and in our case, this vector is passed to a normalization function. Because we are solving for multiple classes, we instruct the network to send the Logits vector to a SoftMax activation function. Then, the resultant output of our SoftMax activation function will be a vector of normalized probabilities.

	NAIP VGG16 Instance 1			NAIP VGG16 Instance 2			NAIP VGG16 Instance 3			NAIP VGG16 Instance 4		
	Rural	Suburban	Urban	Rural	Suburban	Urban	Rural	Suburban	Urban	Rural	Suburban	Urban
Rank 1 Probabilities	0	1.38 E-19	1	2.00 E-25	3.72 E-14	1	0	7.34 E-17	1	9.37 E-36	1.32 E-14	1
Rank 3 Probabilities	0	7.26 E-19	1	1.99 E-25	4.60 E-14	1	0	1.26 E-16	1	2.90 E-35	4.26 E-14	1
Rank 5 Probabilities	0	2.55 E-18	1	2.51 E-25	6.10 E-14	1	0	6.33 E-16	1	8.65 E-36	1.44 E-14	1
Rank 625 Probabilities	0	1.51 E-14	1	5.17 E-22	1.37 E-11	1	2.76 E-37	1.38 E-13	1	1.56 E-32	1.87 E-12	1
Rank 866 Probabilities	0	5.98 E-15	1	1.57 E-19	1.18 E-10	1	1.03 E-34	4.85 E-13	1	2.11 E-30	9.59 E-13	1
Rank 1250 Probabilities	0	5.18 E-06	0.999995	2.12 E-16	6.71 E-05	0.999933	1.46 E-23	0.000257	0.999743	1.25 E-24	7.85 E-06	0.999992
Rank 1 Logits	-112.10	30.40	73.83	-32.01	-6.07	24.86	-35.53	26.42	63.57	-34.67	14.03	45.99
Rank 3 Logits	-113.11	31.54	73.30	-32.16	-5.99	24.72	-35.68	26.67	63.29	-34.50	14.24	45.03
Rank 5 Logits	-112.24	31.41	71.91	-32.14	-5.93	24.50	-35.15	26.73	61.73	-34.62	14.24	46.11
Rank 625 Logits	-100.11	29.18	61.00	-27.99	-3.99	21.02	-30.52	24.05	53.66	-31.41	14.82	41.83
Rank 866 Logits	-89.84	24.98	57.73	-24.58	-4.14	18.72	-28.18	21.72	50.08	-29.36	11.30	38.97
Rank 1250 Logits	-80.46	26.20	38.37	-22.58	3.90	13.51	-19.85	24.46	32.73	-23.16	20.12	31.88

Figure 8 Image showing Urban class imagery, their relative ranks

Pooling – Simonyan and Zisserman designed VGG16 with pooling layers after every few convolutions. Pooling is a kind of subsampling or down-sampling and could be accomplished through many approaches, max, min, average, range, variance, kurtosis, etc. However, most practitioners tend to use max pooling or average pooling. Some of the other approaches just mentioned would be an interesting future course of study. In the case of VGG16, the designers selected to use max pooling, this concept is shown in Figure 9. One of the benefits of pooling is to reduce the computational burden on the network by reducing the number of parameters to assess. Additionally, in the case of VGG16, it allowed larger features to become manifest in later layers as the features became smaller and more sensitive to the small 3x3 kernels. Because these pooled values are derived from the prior convolution layer, they are no trainable parameters directly associated with pooling layers in VGG16.

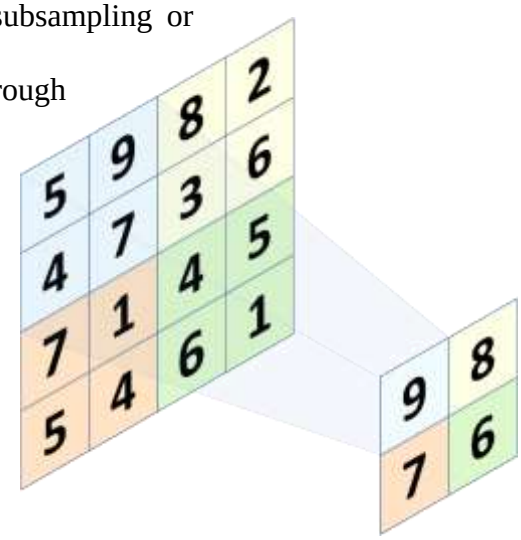


Figure 9 Conceptual Graphic of Max Pooling

Pooling impact on synthetic images - It is important to consider the effects of max pooling on an image as it traverses through the network. In the illustration below, notice that max pooling selects the pixels with the highest values, in this case, it appears as white pixels. After a few iterations, only the pixels with the

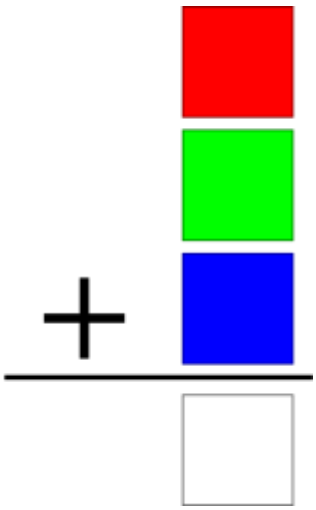


Figure 10 White synthetic images can maximally activate neurons .

highest values remain, and in this case of black stripes, block 4 and block 5 will only see white pixels. Additionally, there is a second item to understand about synthetic images and the network in general. The three channels of Red, Green, Blue (RGB) color images are handled individually and it should be noted that a white image can be considered as an image that contains maximum red, maximum blue, and maximum green. Why is the important? If classes were to be identified primarily based on color, then white synthetic images may present a case where it appears to maximally activate the neurons associated with multiple classes.

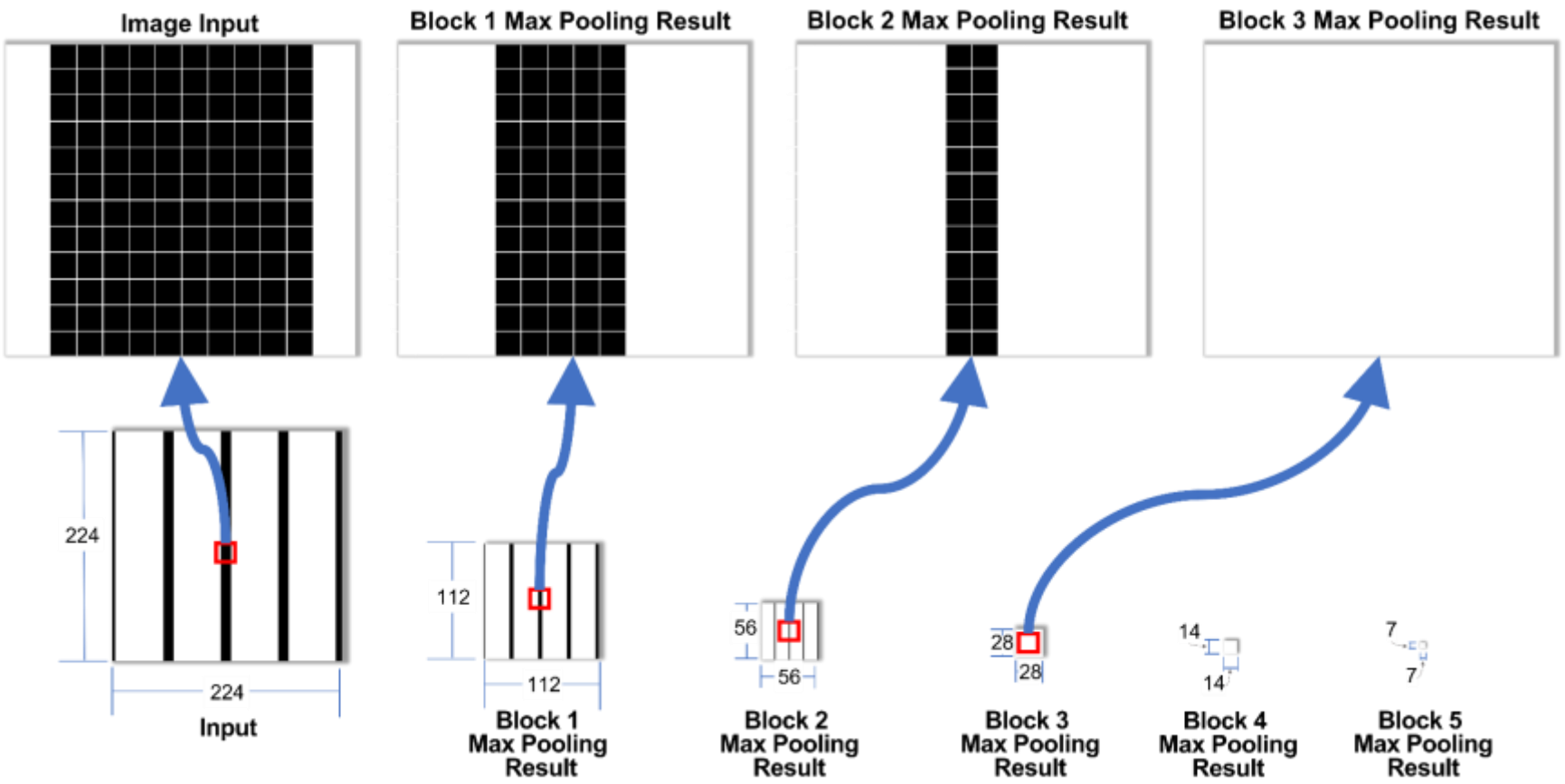


Figure 11 Impact of Max Pooling

Dropout Regularization – In the original VGG models, the authors used dropout regularization between the first two fully connected layers[4]. This helps to prevent overfitting[5]. In our case, we did the same for some of our experiments, but for one large set of experiments to test the feasibility of using a Post Hoc explainability method as a type of distance measure, we selected to train the VGG16 model without drop out regularization to minimize the variance between multiple VGG16 models.

Training the VGG16 - As can be expected, training the VGG16 is an iterative process of minimizing differences between output predictions and given ground truth labels on a training dataset. To do this, values of the kernels in the convolution layers and in the fully connected layers are updated based on "feedback" from an optimization algorithm. For VGG16, the feedback is backpropagation from the resulting gradient descent optimizer. in our case, the optimizer is Adam, (the name Adam is derived from adaptive moment estimation, and it is not ADAM, just Adam.),

VGG's performance is calculated via a loss calculated for a set of trainable parameters (e.g. kernel weights and biases), and updated via the backpropagation feedback, mentioned a moment ago. The direction of the changes to the values are based on Adam's attempt to scale down the gradient descent. If values increase the loss, Adam updates the values to continually descend down the loss 'terrain'.

Learning Rate – The pace of Adam's descent to the minimum, is controlled by the learning rate hyperparameter. Larger learning rates will have Adam take larger

[4] Simonyan, et. al 2014
[5] Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012).

"steps" to find the best route down the loss, but they may be too large and miss the appropriate "path" of just the right trainable parameters. If you have an infinite amount of time, smaller learning rates would be ideal. So, often, learning rates are part of the art of training a neural network. For VGG16, training / learning is a stochastic process, and after every update of the values, the network runs through more dataset to determine adjustments to minimize the loss. This could repeat continually, unless the developer or researcher intervenes, either programmatically or manually, to stop the process. In our case, we added programmatic guidance to minimize overtraining by having the network stop after a certain number of iterations, if the validation loss did not improve.

What is learned, in which part? It's commonly understood that for CNNs, models learn general knowledge in the first layers of the model and more task specific knowledge in the later layers of the model[6]. However, why is this the case? It's surprisingly simple. The early convolution layers train on small features as the image tensor has not yet been reduced via pooling. As the image (tensor) features are progressively pooled, the features that are being convolved, represent larger areas of the original image. As can be seen in the conceptual graphic, the 3x3 kernels in Block 1 train on the size of 224x224, while in Block 5, the 3x3 kernels train on a 14x14 size tensor. That is, the 14x14 tensor represents the whole original image in block 5. You may recall the recent discussion regarding receptive fields. Applied to Block 5, a 7x7 receptive field represents substantially more of the original image, as illustrated in the following conceptual graphic, then a receptive field in Block 1. In the conceptual

[6] Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?." Advances in neural information processing systems 27 (2014).

graphic, the size of the smiley face, after four pooling steps, is very close to 7x7 pixels in Block 5.

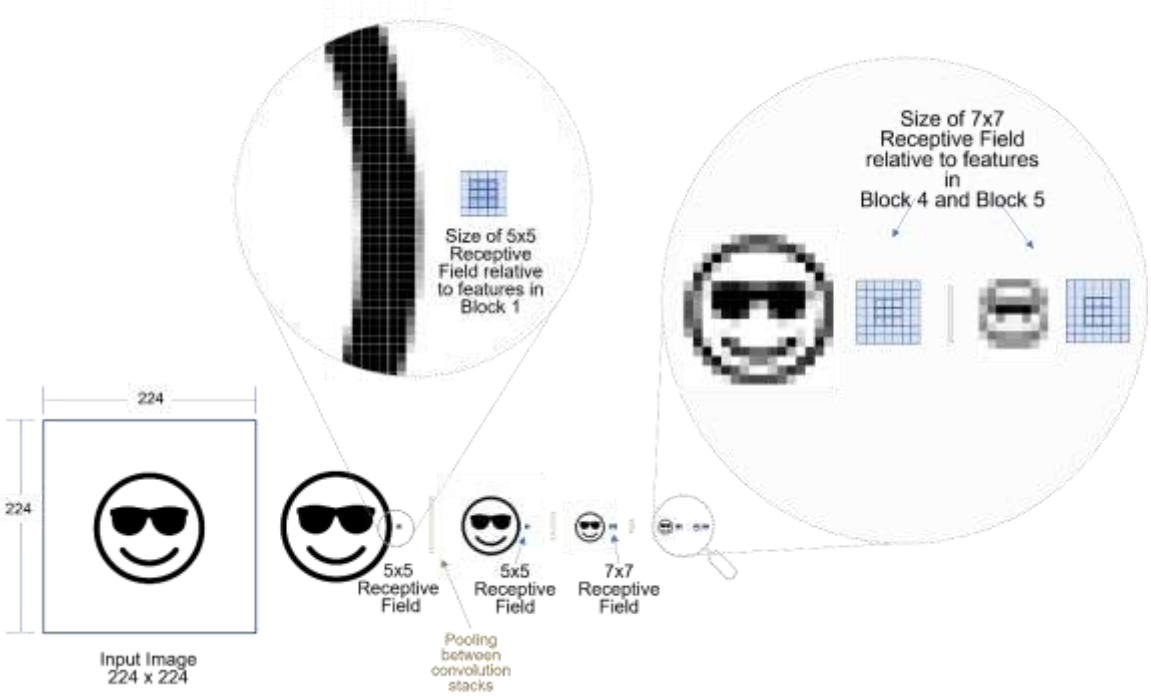


Figure 12 Concept illustrating relative size of receptive field to image representations

Advanced Fine Tuning: Gradual Unfreezing

In addition to typical Transfer Learning experiments, we will be using a variant to better manifest changes in the network. That is, we used a variant of Fine Tuning that has been called Gradual Unfreezing, because it unfreezes the frozen base model. Perhaps 'Gradual Thawing' was already taken. Some of Gradual Unfreezing approaches[7] include layer by layer unfreezing, starting at layers closest to the classifier, and training on only 1

[7] Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." arXiv preprint arXiv:1801.06146 (2018).

epoch, then unfreezing the next layer down and train for only 1 epoch, so forth, Howard and Ruder (2018).

Another variant is to start only training the classifier, then unfreezing one layer closest to the classifier, training for 50 epochs, and pick the best performing model from the 50 epochs. Then re-freeze all the layers except for the layers that are *furthest* from the classifier, train for 50 epochs, pick the best performing model, re-freeze all the layers, except for the next layer closest to the classifier, train for 50 epochs, pick the best, re-freeze and repeat all the way until all the layers have been retrained one at a time. After that, retrain the whole network. [8]

Another approach similar to that described above, but unfreezing in the opposite direction[9], that is unfreezing and fine tuning individual layers closest to the classifier, all the way to the layers that are furthest from the classifier, one layer at a time.

Our approach to Gradual Unfreezing is like the approach just described, unfreezing from the layers closest to the classifier, onward to the layers that are furthest from the classifier. Other researchers have used this approach successfully, too[10]. However, we are unfreezing VGG blocks at a time, as they were meant to operate together.

Often, during fine tuning, when using pre-trained weights from a large model like ImageNet, the classifier block starts with Glorot distribution because the large model

[8] Felbo, Bjarke, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm." arXiv preprint arXiv:1708.00524 (2017).
[9] Wiedemann, Gregor, Eugen Ruppert, Raghav Jindal, and Chris Biemann. "Transfer learning from lda to bilstm-cnn for offensive language detection in twitter." arXiv preprint arXiv:1811.02906 (2018).
[10] White, Gary, Christian Cabrera, Andrei Palade, Fan Li, and Siobhan Clarke. "WasteNet: Waste classification at the edge for smart bins." arXiv preprint arXiv:2006.05873 (2020).

classifier block has significantly more kernels than researchers need, such as we do in our three class experiments. So, researchers tend to lose the benefit of that pre-training in the classifier block.

As our synthetically trained models represent three classes, we will start with those pretrained synthetic values, even in the classifier block. After each iteration, we select the model from the epoch that has the lowest loss during validation and use those values to start the next iteration. For example, the best model of Iteration 1, becomes the model to start Iteration 2. The best model of Iteration 2, becomes the model to start Iteration 3, etc. This way, the learning is preserved.

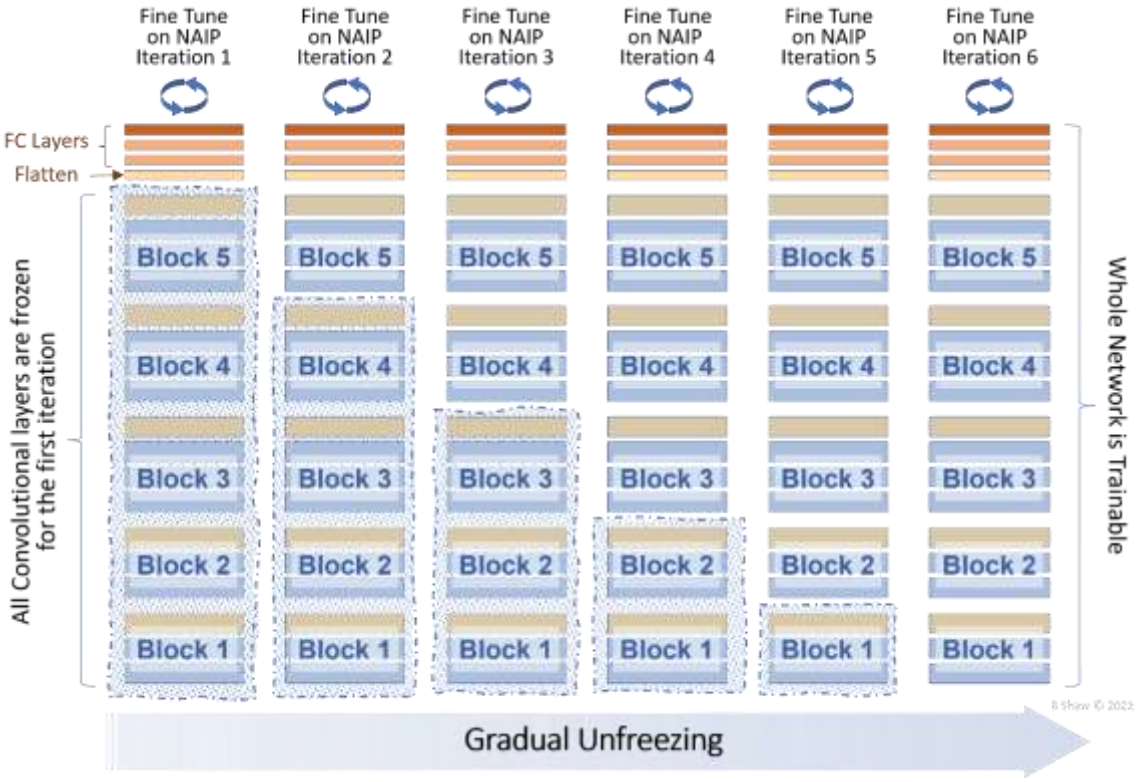


Figure 13 Gradual Unfreezing is block by block

DISCUSSION OF RESEARCH QUESTIONS

In the abstract, we mentioned potential differences in the characteristics of natural photographic image datasets and Remote Sensing. Let's expand this and unpack our research questions. Regarding these differences, there are, for example, differences in data sizes, features and task specifications between the source data sources and the requirements of our target domain, Remote Sensing.

As stated earlier, to address these issues, the goal of this study is to determine if a fairly arbitrary, easily produced, set of synthetic datasets can be developed and used for transfer learning for a typical deep learning task. So, we would want the dataset to reflect attributes of the target dataset.

So, the first research question becomes: **RQ 1: Can we make useful, synthetic datasets using simple tools?** That is, can we make synthetic datasets ***without*** using specialized generative tools such as a Generative Adversarial Network?

Complimenting this, an insightful paper from researchers at Google Brain revealed an issue of Transfer Learning on medical imagery. Specifically, Raghu et al[11]. observed that models trained on medical image datasets did not learn Gabor filters, which were the case when models were trained on ImageNet data, challenging some popular

[11] Raghu, Maithra, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. "Transfusion: Understanding transfer learning for medical imaging." Advances in neural information processing systems 32 (2019).

ideas at the time. That is, a few years prior, Yosinski et. al. (2014) [12]. stated "Modern deep neural networks exhibit a curious phenomenon: when trained on images, they all tend to learn first-layer features that resemble either Gabor filters or color blobs"

It might be noted with a bit of humor, that a coauthor Yosinski was Yoshua Bengio, while a coauthor of Raghu was Samy Bengio, Yoshua's little brother. It is fun to imagine the debates and scuffles when they meet for holidays. "They do make Gabor filters !", "They don't make Gabor filters !", "They do make …"

Besides humor, this provides a driver for our approach. We want to make sure our source synthetic dataset reflects properties of the target synthetic dataset. So, the second research question becomes **RQ 2: What should we use to measure the efficacy of a source synthetic dataset?** If we merely rely on accuracy after transfer learning, the results might not readily give us a direction to change our synthetic dataset and we would become a kind of a human Generative Adversarial Networks (GAN). So, instead, recognizing that this will be an iterative approach, we need to consider how to measure the performance of the network trained on an iteration of a synthetic dataset so that we can glean direction to make changes in the next iteration of our synthetic datasets. We want to be able to train test, and ask simple questions, such as. "Which image attributes resonated with the target dataset?" "Should we use more green or less green in our next iteration of synthetic dataset?", "Which directions should have edges?" "How quickly do those edges transition?", "What sizes features do we need?" etc. With these answers, we should be able to guide our development of the next iteration of the synthetic dataset.

[12] Yosinski, et. al 2014.

LITERATURE SURVEY

Transfer Learning

Regarding generated synthetic data used as source datasets, both Generative Adversarial Networks (GAN) and 3D simulation have been very prevalent in synthetic training approaches. For 3D simulated data, scenes are rendered from custom simulation tools such as Carla (Dosovitskiy et al., 2017[13]), Synscapes (Wrenninge and Unger, 2018[14]), Synthia (Ros et al., 2016[15]), even from video games such as, Grand Theft Auto V (GTA V), (Johnson-Roberson, et. al, 2016[16]), or subject domain specific such as from a fish farming simulator called SimSalma, (M Reiersen, 2018[17]). Tremblay et al. (2018[18]) used synthetic data generated from the game development tool, Unreal Engine 4 (UE4), and blended the synthetic data into background photographs taken from a Flickr photographic dataset. They place 3D simulated objects of interest randomly over

[13] Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An open urban driving simulator." arXiv preprint arXiv:1711.03938 (2017).
[14] Wrenninge, Magnus, and Jonas Unger. "Synscapes: A photorealistic synthetic dataset for street scene parsing." arXiv preprint arXiv:1810.08705 (2018).
[15] Ros, German, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3234-3243. 2016.
[16] Johnson-Roberson, Matthew, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?." arXiv preprint arXiv:1610.01983 (2016).
[17] Reiersen, Magnus. "Deep Visual Domain Adaptation:-From Synthetic Data to the Real World." Master's thesis, NTNU, 2018.
[18] Tremblay, Jonathan, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 969-977. 2018.

background photographs. Also, the team add objects of disinterest, (distractors), in the images randomly. Denninger et. al. (2020) [19]developed BlenderProc, built on the open source Blender tool suite, to provide configurable pipeline for procedurally generating rendering photorealistic 3D images and scenes to be used for training deep learning models. Nowruzi et al. (2019)[20] conducted a meta-study investigating a series of 3D simulated synthetic datasets to show that training on 3D simulated synthetic data and fine tuning on real data yield better performance than training on a mixed real-synthetic dataset. Prakash et. al. (2019) used the UE4 game [21]engine to develop the synthetic data with the intent to create the synthetic scenes and the corresponding objects of interest as well as objects of disinterest with the same probability distribution as a real scene. Hinterstoisser, et. al. (2019) [22]generated synthetic data by adding Gaussian noise to the object of interest and Gaussian blurring the object edges before compositing the image over a background image.

Regarding GAN, researchers in the medical fields are using adversarial networks to generate data because the paucity of images and data of anomalous conditions. For

[19] Denninger, Maximilian, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomas Hodan, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi. "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering." In Robotics: Science and Systems (RSS) Workshops, vol. 2, no. 3, p. 7. 2020.

[20] Nowruzi, Farzan Erlik, Prince Kapoor, Dhanvin Kolhatkar, Fahed Al Hassanat, Robert Laganiere, and Julien Rebut. "How much real data do we actually need: Analyzing object detection performance using synthetic and real data." arXiv preprint arXiv:1907.07061 (2019).

[21] Prakash, Aayush, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. "Structured domain randomization: Bridging the reality gap by context-aware synthetic data." In 2019 International Conference on Robotics and Automation (ICRA), pp. 7249-7255. IEEE, 2019.

[22] Hinterstoisser, Stefan, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. "On pre-trained image features and synthetic images for deep learning." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 0-0. 2018.

example, Kandala, et. al (2018) [23] used a GAN to generate images with lesions such that the overall severity level of the retinal lesions can be controlled. Correspondingly, for other fields, such as Remote Sensing, researchers are leveraging adversarial networks to expand or create training datasets. For example, Liu et. al (2020)[24] extended the approach of randomizing source training data by refining the synthetic images with a GAN-based image translator to generate content-rich, realistic synthetic images with ground truth annotations.

More recently, in Neyshabur et. al's (2020)[25] paper "*What is being transferred in transfer learning?",* the team investigates Transfer Learning between a model trained on ImageNet and multiple target domains from DomainNet, including clip art images, (DomainNet Clipart) and photographic images, (DomainNet Real). In one of the aspects of that study, they noted that test accuracy of the model finetuned on DomainNet Real achieved about a test accuracy of about 76%, while the model finetuned on DomainNet Clipart achieve a test accuracy of about 73%. The clipart test accuracy of 73% is surprisingly close to the results of transfer learning between two sets of photographic images, ImageNet and DomainNet Real.

In a very different knowledge domain, a language model of written English, Chiang, et al (2020)[26], discovered that truly synthetic data, (i.e., generated, not-even-a-Human language data), was surprisingly useful as a pre-training language model for

[23] Pujitha Appan, K., and Jayanthi Sivaswamy. "Retinal Image Synthesis for CAD development." (2018).
[24] Liu, Weixing, Jun Liu, and Bin Luo. "Can Synthetic Data Improve Object Detection Results for Remote Sensing Images?." arXiv preprint arXiv:2006.05015 (2020).
[25] Neyshabur, Behnam, Hanie Sedghi, and Chiyuan Zhang. "What is being transferred in transfer learning?." arXiv preprint arXiv:2008.11687 (2020).
[26] Chiang, Cheng-Han, and Hung-yi Lee. "Pre-Training a Language Model Without Human Language." arXiv preprint arXiv:2012.11995 (2020).

downstream English language tasks. However, the authors used English's uni-gram distribution for the training of that non-human language dataset. It may indicate that uni-gram distribution, or the synthetic dataset's other feature(s) distribution which could be similar to English's feature distribution, was a driver in the relatively strong performance of the artificial dataset.

Is It Really Trained *Only* on Synthetic Data?

In a surprising number of papers, authors rely on neural network architectures that are pre-trained on ImageNet, even *while the intent of the papers* are to show *the benefit of training on synthetic data*. This is particularly true when their architecture is a two-stage segmentation architecture and they rely on a pretrained CNN to help identify the candidate mask areas for their (stage one) object detector, and then the same CNN as a feature extractor (stage two) for classifications.

Although Prakash et. al. (2019) [27]state they train their object detector on synthetic data, they are using a feature detector pre-trained on ImageNet, then retrained on the synthetic data. … "We demonstrate the power of [structured domain randomization] SDR for the problem of 2D bounding box car detection, achieving competitive results on *real data after* ***training only on synthetic data***." However, in their experimental discussion they state " Resnet V1 **pretrained on ImageNet** was used as the feature extractor"

- In their mistitled paper, ***Object Detection Using Deep CNNs Trained on Synthetic Images***, Rajpura, et. al (2017)[28], reveal they used a GoogleNet instance pre-trained on

[27] Prakash et. al. (2019)
[28] Rajpura, Param S., Hristo Bojinov, and Ravi S. Hegde. "Object detection using deep CNNs trained on synthetic images." arXiv preprint arXiv:1706.06782 (2017).

ImageNet, as they describe their experimental set up. "…For our experiments, we use **pre-trained weights on ImageNet** to initialize the [Fully Convolutional Network] FCN network which has earlier been helpful for transfer learning…"

- Tremblay et al. (2018)[29] conducted three experiments with three different neural network architectures, but in all three they used networks pretrained on ImageNet as feature extractors, specific two architectures using Inception-Resnet V2 **pretrained on ImageNet** and one architecture using Resnet101 pretrained on ImageNet.
- Liu and Mildner (2020) [30]mistitled thesis "***Training Deep Neural Networks on Synthetic Data***" state that they intend to "use synthetic data generated by a computer for training" and to "investigate this idea by training the object detector YOLOv3 on synthetic images., then "study the difference between models trained on real and models trained on synthetic data." However, for all of their models using synthetic data, they "Initialized with the Darknet-53 backbone".
    - Unfortunately, the authors leave it up to the reader to know that Darknet 53 backbone is a set of configuration parameters and weights **developed by pretraining on ImageNet**. Darknet 53 and other Darknet backbones can be found here: https://pjreddie.com/darknet/imagenet/. Note that the description of the various Darknet models at the aforementioned site states "Here are a variety of pre-trained models for ImageNet classification."

[29] Tremblay et al. (2018)
[30] Liu, Tony, and Arvid Mildner. "Training Deep Neural Networks on Synthetic Data." LU-CS-EX (2020).

- Linder, et. al (2019) [31]intended to show the benefits of training a network "**from scratch**" on their synthetic dataset. However, the network that they use is YOLOv3, which was pretrained on ImageNet. "…We use the MxNet implementation of the YOLOv3 detector (with DarkNet53 **pretrained on ImageNet**) for our experiments. … "
- Hinterstoisser, et. al. (2019) [32] stated that they were able to " train state-of-the-art object detectors purely on synthetic data." However, contrary to their claim, those CNNs were originally trained on real data by the developers of the original CNNs. That is, Hinterstoisser, et. al. used the parameters from the CNNs trained on real data as the starting parameters for the training with the synthetic data.
    - "… Therefore, by freezing the weights of feature extractor pre-trained on real data and by only adapting the weights of the remaining layers during training, we are able *to train state-of-the-art object detectors purely on synthetic data*."
        - "… While the 'frozen' parts are taken according to [1], by training Inception-Resnet and Resnet101 on a classification task on the ImageNet-CLs dataset. We freeze Inception-Resnet (v2) after the repeated use of block17 and right before layer Mixed 7a and Resnet101 after block3. All other remaining parts of the networks are not 'frozen', meaning *their weights are free to adapt when we train the detector on synthetic images*. " [Italics are mine: meaning, the

[31] Linder, Timm, MJ Hernandez Leon, Narunas Vaskevicius, and Kai O. Arras. "Towards training person detectors for mobile robots using synthetically generated RGB-D data." In Computer Vision and Pattern Recognition (CVPR) 2019 Workshop on 3D Scene Generation. 2019.

[32] Hinterstoisser, Stefan, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. "On pre-trained image features and synthetic images for deep learning." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 0-0. 2018.

later layers, were pre-trained on real data, but then were fine tuned using the synthetic data – Specifically, “free to adapt” is not the same thing as random initialization.]

- As an alternative to high-fidelity 3D synthetic images, Tobin et al. (2017) intended to introduce sufficient source domain randomization by generating sufficient synthetic data with sufficient variation that the trained neural network viewed real-world data as just another variation. Specifically, they intended to use domain randomization and train a neural network to estimate the 3D world position of various shape-based objects with respect to a robotic arm fixed to a table. However, they used weights obtained by pretraining on real data
    - "…For the majority of our experiments, we use weights obtained by pretraining on ImageNet to initialize the convolutional layers, which we hypothesized would be essential to achieving transfer…."

Concepts and Testing with Concept Activation Vectors (TCAV)

What do we measure? Of course, we’ll measure accuracy, but we are also concerned that hidden features, or representations learned by a pre-trained model should match those of model trained on a target data source. So, we need to find a way to measure what is learned. In our case, we landed on measuring learned ‘concepts’.

Why concepts? Because understanding and explaining decisions made by deep neural networks are important to validate decisions during development of the neural network and afterward, when users depend on the neural network to provide support for decisions. This is, the driver for Explainable Artificial Intelligence efforts. Frequently,

neural networks are viewed as black boxes that provide a likelihood of correlation to a class but do not provide any interpretable justification that is meaningful. For example, a neural network may identify that a small object in a Remote Sensing image is a large transport vehicle, but it doesn't provide the rationale to explain the object's correlation to the large transport vehicle class. To address the limitation, some researchers have pursued understanding of the correlation in the form of individual features or sets of features of an image, often by removing or altering the features to demonstrate their importance to the correlation between the object and the class.

As part of the effort to provide more effective explanations for human understanding, some researchers are pursuing explanations with more easily understandable concepts. These concept-based explanations are intended to reveal the neural network's behavior in a way that is understandable to humans by revealing the use of the concepts in decisions made by the network. For example, a particular urban land use class may show high sensitivity to sets of orthogonal lines, while a particular rural land class in a temperate climate zone may show high sensitivity to a yellow-green color. In this case, those specific image attributes of "sets of orthogonal lines" and "yellow-green color" are some of the 'concepts' used by the neural network to make correlations between an image and a class. Although it is beyond the scope of this work to explore all the approaches to explain neural network decisions, it should be noted that there are multiple approaches, they are comprehensive, and the aforementioned concept-based-explanations are but one approach.

Testing with Concept Activation Vectors, (TCAV) introduced by Kim et. al. (2018) [33] assesses the sensitivity of labeled concepts to specific classes of a trained model. Their TCAV approach trained linear classifiers to derive concept activation vectors for each example of a labeled concept. The sensitivity of the example to the classes is based on the directional derivates of concept activation vectors.

We use the term 'concept' in line Kim who stated there is a difficulty in providing a precise definition of concept. However, we wish to provide a definition to our particular case for ease of understanding in the context of this work. In our case of examining Remote Sensing images, a 'concept' is defined as any image attribute, whether a simple and low-level attribute, such as an 'edge', or a 'color', to more abstract, higher-level attributes as 'street' or 'skyscraper' or 'foliage'. Moreover, this includes those image attributes that require a contextual understanding of the image data such as 'well-worn footpath' or 'dry lakebed' or '18-hole golf course'. Although the term 'image attribute' would be perfectly useable in our case, we select to use 'concept' so that readers can track with other concept-based-approach literature.

TCAV has been expanded, augmented, and applied in various disciplines. For example, Clough et. al. (2019) [34] classified coronary artery disease using key biomarker concepts. Importantly, their approach provided better interpretability using known biomarkers because the model's predictions were placed in the context of current medical knowledge and clinical decision-making guidelines. TCAVs were expanded to support

[33] Kim, Been, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, and Fernanda Viegas. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)." In International conference on machine learning, pp. 2668-2677. PMLR, 2018.
[34] Clough, James R., Ilkay Oksuz, Esther Puyol-Antón, Bram Ruijsink, Andrew P. King, and Julia A. Schnabel. "Global and local interpretability for cardiac MRI classification." In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 656-664. Springer, Cham, 2019.

Recurrent Neural Networks (RNN)s by [35] is the support of sequential modeling of adverse outcomes in Electronic Health Records. Some studies illuminated the importance of selecting counter-concepts, where [36] noted for their Radiology requirements, counter-concepts should be similar to concepts, likewise for Meteorology uses where counter-concepts needed to have similar image histograms as concepts [37].

As has been noted in the literature [38] this TCAV approach following requires the user to know a priori, the concepts to test. Consequently, using TCAV in Remote Sensing requires knowledge of image analysis, knowledge of the remotely sensed targets, as well as knowledge of deep learning. Others have addressed this slightly, such as Ghorbani et. al.'s (2019) Automatic Concept-based Explanations [39] , to address the issues. That is, to expand its applicability, Ghorbani et. al. automated the development of concepts by segmenting training or testing images, clustering images, and using the segmented images as concepts. To answer the question if a particular set of concepts is in explaining a model's prediction behavior is enough, Yeh (2019) [40] proposed a discovery method to infer a complete set of concepts. This may be a useful follow on.

[35] Mincu, Diana, Eric Loreaux, Shaobo Hou, Sebastien Baur, Ivan Protsyuk, Martin Seneviratne, Anne Mottram, Nenad Tomasev, Alan Karthikesalingam, and Jessica Schrouff. "Concept-based model explanations for Electronic Health Records." In Proceedings of the Conference on Health, Inference, and Learning, pp. 36-46. 2021.
[36] Reyes, Mauricio, Raphael Meier, Sérgio Pereira, Carlos A. Silva, Fried-Michael Dahlweid, Hendrik von Tengg-Kobligk, Ronald M. Summers, and Roland Wiest. "On the interpretability of artificial intelligence in Radiology: Challenges and Opportunities." Radiology: Artificial Intelligence 2, no. 3 (2020): e190043.
[37] Sprague, Conner, Eric B. Wendoloski, and Ingrid Guch. "Interpretable AI for Deep Learning− Based Meteorological Applications." In 99th American Meteorological Society Annual Meeting. AMS, 2019.
[38] Ghorbani, Amirata, James Wexler, James Zou, and Been Kim. "Towards automatic concept-based explanations." arXiv preprint arXiv:1902.03129 (2019).
[39] Ghorbani, et. al. 2019
[40] Yeh, Chih-Kuan, Been Kim, Sercan O. Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. "On completeness-aware concept-based explanations in deep neural networks." arXiv preprint arXiv:1910.07969 (2019).

METHOD

Planning an Iterative Approach

Our original approach was to iteratively develop synthetic datasets based on the results of the transfer learning tasks. The performance of the network after transfer learning was to be measured to guide the refinement of the next candidate source dataset. That is, each dataset would be iteratively developed to close the expected performance gap between VGG16 trained on only the Synthetic Datasets and VGG16 trained National Agriculture Imagery Program (NAIP) data.

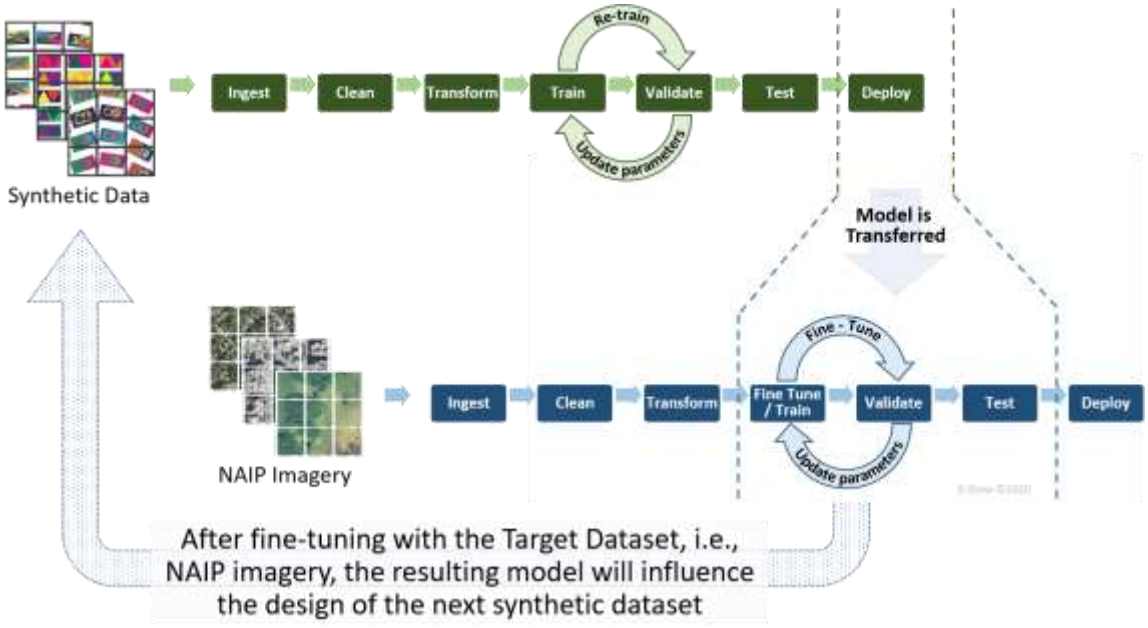


Figure 14 Our workflow to iteratively update models based on performance of fine tuned VGG

You can think of this as similar to a human-in-the-middle approach to a GAN. Correspondingly, each source dataset should be viewed merely as candidates, like candidate filaments in Thomas Edison's experiments.

Gradual Unfreezing

To retain the learning throughout our approach, we will use a controlled gradual unfreezing of the network. Since our synthetically trained models represent three classes, we will start with those values in the classifier block. After each iteration, we select the model from the epoch that has the lowest loss during validation and use those values to start the next iteration. For example, the best model of Iteration 1, becomes the model to start iteration 2. The best model of iteration 2, becomes the model to start iteration 3, etc.

This way, the learning is preserved. To show this concept, attached are screen captures of the TensorFlow code. TensorFlow requires creating a new model, then importing the weights from the source model, in our case it is the best model, with the lowest loss, from the prior iteration of gradual thawing / gradual unfreezing. How do we pick the best model? We save every epoch. So, if we expect to run 100 epochs, we save 100 models, and use the best performing model. Fortunately, there were some wise programmers who added a 'patience' attribute that allowed for early stopping should our model stop improving. Once the best model was selected, the following graphic shows TensorFlow's mechanism to import weights from the prior iteration.

```
In [27]:
########################
### Hand_Built_Models ###
########################

#'''
#### FIRST SET OF CORRESPONDING CONVOLUTIONAL LAYERS #####
new_model.layers[0].set_weights(Original_Model.layers[0].get_weights())
new_model.layers[1].set_weights(Original_Model.layers[1].get_weights())
new_model.layers[2].set_weights(Original_Model.layers[2].get_weights())
new_model.layers[3].set_weights(Original_Model.layers[3].get_weights())
new_model.layers[4].set_weights(Original_Model.layers[4].get_weights())
new_model.layers[5].set_weights(Original_Model.layers[5].get_weights())
new_model.layers[6].set_weights(Original_Model.layers[6].get_weights())
new_model.layers[7].set_weights(Original_Model.layers[7].get_weights())
new_model.layers[8].set_weights(Original_Model.layers[8].get_weights())
new_model.layers[9].set_weights(Original_Model.layers[9].get_weights())
new_model.layers[10].set_weights(Original_Model.layers[10].get_weights())
new_model.layers[11].set_weights(Original_Model.layers[11].get_weights())
new_model.layers[12].set_weights(Original_Model.layers[12].get_weights())
new_model.layers[13].set_weights(Original_Model.layers[13].get_weights())
new_model.layers[14].set_weights(Original_Model.layers[14].get_weights())
new_model.layers[15].set_weights(Original_Model.layers[15].get_weights())
new_model.layers[16].set_weights(Original_Model.layers[16].get_weights())
new_model.layers[17].set_weights(Original_Model.layers[17].get_weights())


#### FLATTEN #####
new_model.layers[18].set_weights(Original_Model.layers[18].get_weights())

#### FULLY CONNECTED LAYERS #####
new_model.layers[19].set_weights(Original_Model.layers[19].get_weights())
new_model.layers[20].set_weights(Original_Model.layers[20].get_weights())
new_model.layers[21].set_weights(Original_Model.layers[21].get_weights())

#'''


right_now = datetime.today()
right_now = right_now.strftime("%Y%m%d__%H%M")
print(right_now)

20220214__1129

In [28]:
```

Figure 15 Importing weights from the prior iteration's best model

After the new model has the weights from the prior iteration, we freeze the layers in accordance with iteration we are conducting In the case of the image below, we have already trained the classifier block, and after we imported the best version, we freeze the layers from block 4 down, and allow block 5 and the classifier block to fine tune on the National Agriculture Imagery Program (NAIP) target data.

```
2022-02-14__1129
```

```
In [28]:
# After transfering weights from best model from Gradual Thaw step,
# freeze next iterations of bottom Layers

# 0:17 Freezes everything, except decision block
# 0:14 Freezes everything, except block 5 and up
# 0:10 Freezes everything, except block 4 and up
# 0:6 Freezes everything, except block 3 and up
# 0:3 Freezes everything, except block 2 and up

# Freeze Bottom Layers

for layer in new_model.layers[0:14]:
    layer.trainable = False
```

```
In [29]:
for i in range(len(new_model.layers)):
    print(i,"\t", new_model.layers[i].trainable, "\t", new_model.layers[i].name
```

```
0       False   HC_v1_TL_NAIP_block1_conv1
1       False   HC_v1_TL_NAIP_block1_conv2
2       False   HC_v1_TL_NAIP_block1_pool
3       False   HC_v1_TL_NAIP_block2_conv1
4       False   HC_v1_TL_NAIP_block2_conv2
5       False   HC_v1_TL_NAIP_block2_pool
6       False   HC_v1_TL_NAIP_block3_conv1
7       False   HC_v1_TL_NAIP_block3_conv2
8       False   HC_v1_TL_NAIP_block3_conv3
9       False   HC_v1_TL_NAIP_block3_pool
10      False   HC_v1_TL_NAIP_block4_conv1
11      False   HC_v1_TL_NAIP_block4_conv2
12      False   HC_v1_TL_NAIP_block4_conv3
13      False   HC_v1_TL_NAIP_block4_pool
14      True    HC_v1_TL_NAIP_block5_conv1
15      True    HC_v1_TL_NAIP_block5_conv2
16      True    HC_v1_TL_NAIP_block5_conv3
17      True    HC_v1_TL_NAIP_block5_pool
18      True    HC_v1_TL_NAIP_flatten
19      True    HC_v1_TL_NAIP_fc1
20      True    HC_v1_TL_NAIP_fc2
21      True    HC_v1_TL_NAIP_predict
```

Figure 16 Freeze layers for another iteration of Gradual Unfreezing

Target Dataset

USDA NAIP raster imagery was acquired from NAIP's 2018 dataset. The default spectral resolution is visible light, natural color (Red, Green and Blue), with a radiometric resolution of 8-bit pixels, per channel, represent brightness values 0 – 255. According to NAIP documentation, the imagery is digitized at a one-meter ground sample distance

(GSD). The horizontal accuracy matches within six meters of photo-identifiable ground control points.

The target domain dataset of 7000 images per class, were drawn from NAIP 2018 raster imagery, specifically Compressed County Mosaics (CCM). MrSID files were downloaded from the Geospatial Data Gateway (GDG), https://gdg.sc.egov.usda.gov/GDGHome.aspx. The MrSID files were converted to TIFF images using Extensis' GeoViewer tool, Version 9.0.3.4428 at four-meter GSD, (25% of full imagery resolution). The files were classified Urban, Suburban, and Rural, in accordance with their location. Specifically, areas in Washington, DC (~ 38.89, - 77.03) as Urban, areas in Montgomery, MD (~ 38.99, - 77.13) as Suburban, and areas in Price County, WI, (~ 45.81, - 90.26) as Rural. The NAIP imagery is cropped to match pre-trained VGG16 model requirements, 224x224 pixels. The 3 channel RGB was maintained at 8 bit color.

Training, Testing, and Validation Datasets

Then 80% (5000 images per class) of the dataset was set aside for training and validation (during back propagation), and 20% (1250 images per class) was set aside for post training testing.

- **Training Dataset:** – 4000 images per class
- **Validation:** – 1000 images per class
- **Testing Dataset:** – 1250 images per class

Data Augmentation.

Data augmentation was constrained to only translation of the images to preserve latent concepts and only Urban images data were augmented in this manner. Below are some examples of the simple translations from the Hirshhorn Museum.

Figure 17 Example of Simple Translation used for Image Augmentation

Curation

Prior to building our 35 VGG16 networks, the data was tightly curated to ease the learning of the neural networks and minimize misclassification. The curation became a potential action when early training arose due to undocumented syntax changes in Keras. One of the ideas to help the networks train was to tightly curate the data. Ultimately, starting with 7000 images per the Urban class, curation was accomplished through exposure to another neural network, a Mobilenet Neural Network hosted by Google's TensorFlow team. Outside of this task to identify images typically misclassified, Mobilenet was not used in our study. Specifically, we used a variant of this tutorial, https://www.tensorflow.org/hub/tutorials/image_feature_vector to identify typically misclassified images, and used a Mobilenet neural network found on that tutorial.

For example, the curation induced removing nearly all park areas within Washington, DC as well as other areas with predominantly green spaces. Likewise, high population density areas in our Suburban data set of Montgomery County were removed as they were essentially Urban, just not located in Washington, DC.

After curation, almost 6300 images were available per class. Specifically, Urban required the most curation, which reduced the dataset to just above 6300 for Urban. Suburban had fewer images to remove and Rural had only a few images to that could be cause for misclassification, but to balance the datasets, imagery was randomly removed to match the class size of Urban. To make the mental math easy, we randomly discarded the images so that we had 6250.

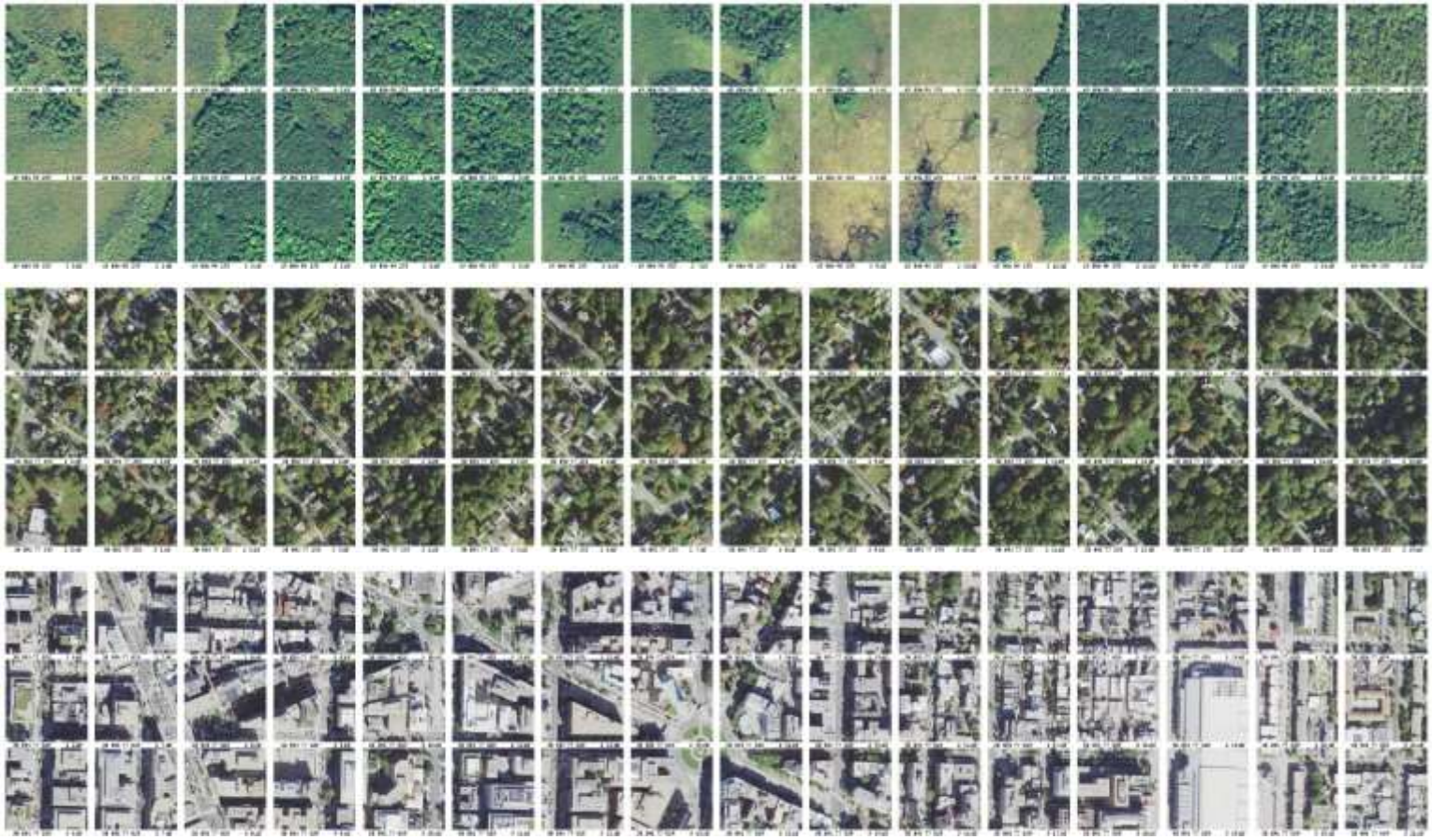
Perhaps, over curation: From hindsight, the Mobilenet neural network used for curation may have been more restrictive than the VGG16 networks would have been. That is, by the time we tested VGG16 models on the segregated test data, all 35 performed perfectly, all had 100% accuracy. It was so concerning, we had to mis-label a few samples just to ensure our inference was acting correctly. However, ultimately, it became a happy mistake as it provided a solid testbed to use in our experiments.

Stratification

Early experiments with VGG11 revealed that Keras selected validation data from the last set of files within a dataset. So, to ensure datasets were representing an even distribution of the imagery, data was stratified by use of a simple prefix (letters A through J) applied to every 10^{th} image in the original dataset. That is, the first, eleventh, twenty first etc image would have a prefix of A, the second, twelfth, twenty second image with be

assigned a prefix of B, and so forth. So, in this manner imagery from across the dataset would be appended with the prefixes.

Example Imagery from NAIP



Source Synthetic Datasets

As shown in Figure 18, synthetic datasets were developed to emphasize different elements that could be leaned by the neural networks, including colors, edges, and patterns. All were developed using commonly available tools and layers of simple shapes. For Gradual Unfreezing experiments, two graphical versions of Rural, Suburban, and Urban imagery were developed, identified as Graphical R S U Version 1 and Version 2. They differ in the Urban class to examine the impact that changes in the synthetic Urban features have on accuracy and TCAV results.

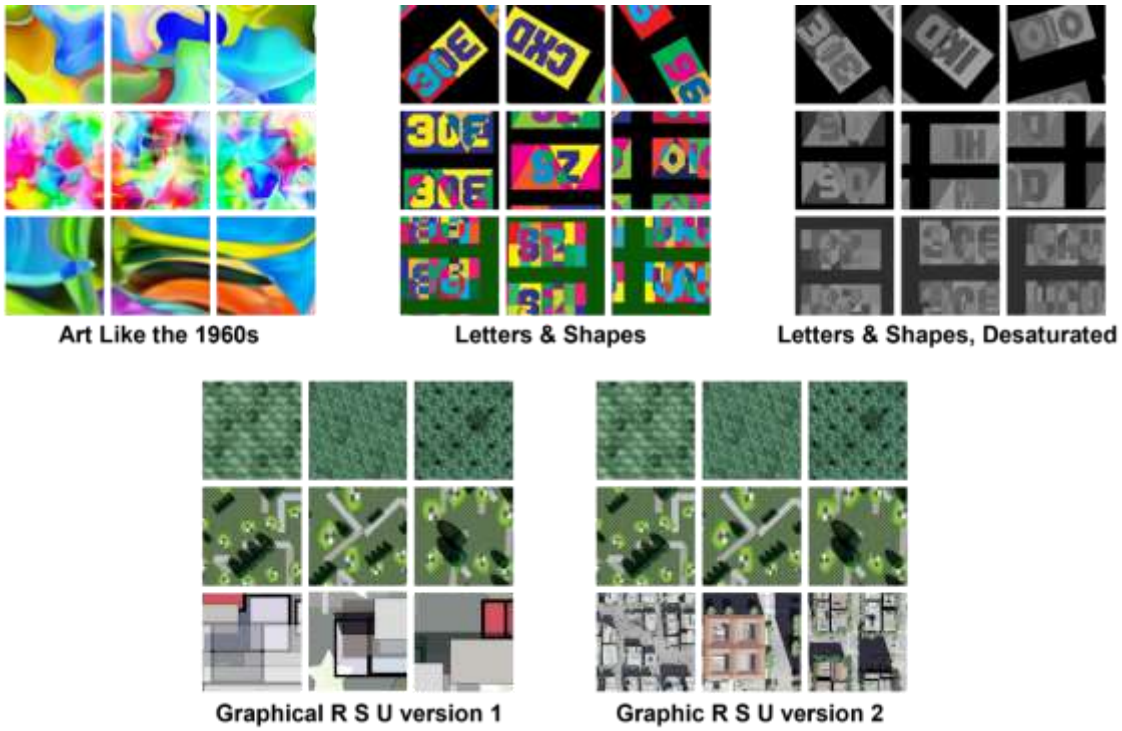


Figure 18 Synthetic Datasets used in Transfer Learning and Gradual Unfreezing

We developed VGG16 models trained on a variety of this simple graphical imagery. The synthetic imagery of note included layers of features derived from the target dataset, including one variant of imagery that fairly successfully mimicked the appearance of the Urban class.

Concepts Datasets

Separate from the synthetic dataset, we measure what is learned in the networks, by testing the network with concept images through the TCAV approach. For example, the concepts included simple shape variants, such as rectangles, color blobs, color-washes, stripes, and vertices. The concept images were generated to match VGG16 model requirements, 224x224 pixels, 3 channel RGB, 8 bit color.

Many of the shapes included rotational variants, gray level variants and/or color variants as separate concepts. To provide variety in the training images, the concepts were translated horizontally and vertically by a few pixels per image.

Additionally, to ensure the TCAV process was responsive as believed, we used example images from the training dataset of the Remote Sensing images as image

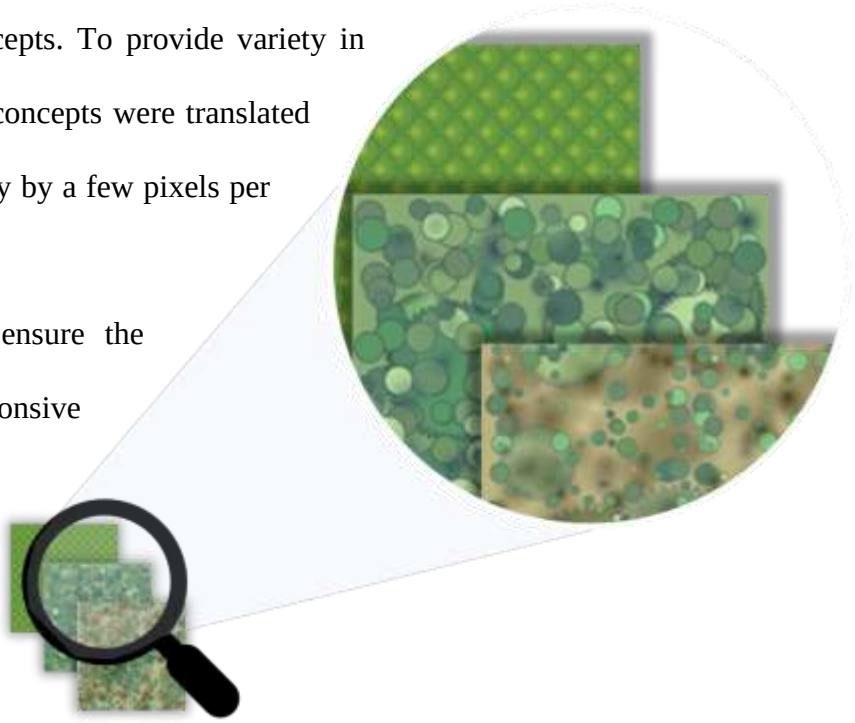


Figure 19 Graphic Concepts to induce response from Rural Class

concepts. That is, we ran TCAV tests of Rural, Suburban, and Urban images as concepts with the expectation that each of those concepts would show sensitivity to their corresponding class.

Note that some of the images have small features in attempts to gain a response from the Rural class. They were designed based on feature size and colors derived from the Rural dataset. To do so, random selection of Rural images were analyzed and measured to determine predominant colors and feature sizes.

Other concept images were designed to test the classes against color sensitivity. So, in addition to black and white stripes and gray scale arc variants, there were red, yellow, green, blue-green, blue, and violet versions of the same. To be specific, the red was RGB of (255,0,0), the green was RGB of (0,255,0) and the blue was RGB of (0,0,255). The three remaining colors were the offset of 60° in the Hue Saturation Brightness (HSB) color space to produce RGB (255,255,0) for yellow, RGB (0,255,255) for blue-green, and (255,0,255) for violet.

Both the arcs and two sets of diagonal stripes were designed to test sensitivity to edges of varying levels of black, (k = 10, 30, 70, and 90), while the vertical stripes were designed to test against varying thickness of black stripes (10 pixels, 20 pixels, 30 pixels, and 40 pixels wide) , knowing that VGG16 used max pooling, instead of average pooling. So, there were was a method to the madness.

Random Counter-Concepts Dataset

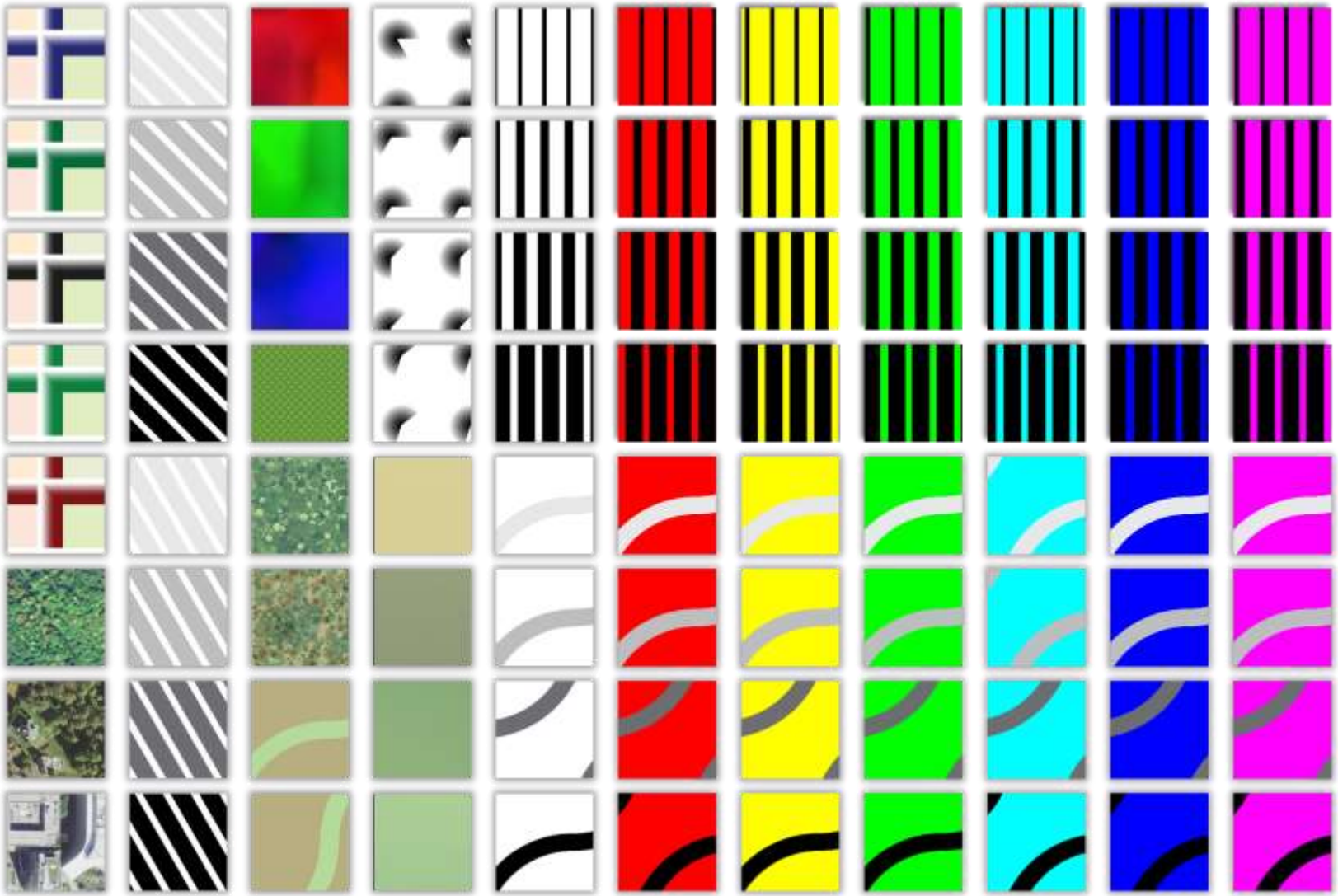
We used images randomly selected from the counter classes as the counter-concepts for each target class. For example, for the target class of Urban, we used only

randomly selected images from the Rural and Suburban classes as the counter-concepts. Likewise, for Rural as target class, we used randomly selected images from Urban and Suburban as counter classes, and for the Suburban target class, we used randomly selected images from Rural and Urban as counter-concepts.

This approach is worth clarifying as Soni, et. al.[41] have attempted variations on the counter-concepts. Although it was explained by Kim et. al.[42], that their counter examples were randomly selected from ImageNet, it may not have been clear that those were images from the model's training dataset, except the images were not from the target class. We too originally attempted to use truly random data sets, but it showed much wider variance in the TCAV results, and we abandoned that approach to generate counter-concepts.

[41] Soni, Rahul, Naresh Shah, Chua Tat Seng, and Jimmy D. Moore. "Adversarial TCAV--Robust and Effective Interpretation of Intermediate Layers in Neural Networks." arXiv preprint arXiv:2002.03549 (2020).
[42] Kim, et. al. 2018

Figure 20 These represent the 88 concepts used in TCAV testing.



Neural Network Models

We used a series of thirty-five (35) VGG16 models, each initialized with same hyperparameters, including seed value to minimize variance of the training. We did not include a dropout layer to minimize the differences in the models. We trained the VGG16 models only on Remote Sensing data, all layers used the default Glorot initialization. No layers used any pre-populated kernel weight or bias values.

Because we curated the data with Mobilenet, each of the thirty-five (35) trained VGG16 models quickly achieved convergence of 100% validation accuracy (1-5 epochs) with the Adam optimizer (learning rate of 10^{-4}). Nevertheless, we continued to train to 25 epochs to ensure the weights and biases were relatively stable.

Hardware

We processed the models on a system using a single Nvidia 2080 TI with 4,352 arithmetic logic units (ALU). In a study last year, the large ALU count allowed me to accelerate the real time training such that each training epoch only took 32 seconds.

Leveraging TCAV as a Proxy for the Changes in each Neural Network

As mentioned, Concept Datasets were developed starting with basic, rudimentary graphic shapes with the expectation that they would not be very sensitive to the complex images of the Remote Sensing classes. During the initial TCAV testing we were surprised to find that some of the concepts were sensitive to the Remote Sensing classes. As such, we developed additional rudimentary graphic shapes, and some of those sets

showed sensitivity. Please note, as indicated by Yeh, et. al,[43] these concepts would not be considered a complete set of concepts necessary to sufficiently explain a model.

Importantly, the TCAV approach includes quantitative validation of the sensitivity of a concept to a class by conducting a series of two-sided t-tests. The series of t-tests are between the Concept Activation Vectors (CAVs) generated from the image concepts of interests and CAVs generated from random selected images as counterexamples. We conducted a series of Welch's T-Test to accommodate the possibility that we may have unequal variances should the process ingest differing number of example concepts and/or random concepts. We conducted TCAV test of the graphical concept sets and of the concepts sets from the Remote Sensing images against the thirty-five (35) identically trained VGG16 models.

TCAV as Distance Measures

In addition to using TCAV to measure specific concept sensitivity per class, we used the TCAV results as dimensions of a tensor / vector representing each network. In our case, we used 88 TCAV results of the Predict layer of the 35 VGG16 networks trained only on NAIP and 88 TCAV results of the predict layers of those trained on synthetic sources, who then were retrained on our target domain underwent transfer learning. We also used the Average, Median, and Mode of TCAV values of the 35 VGG16 networks trained on NAIP data.

The intuition for this approach is that the VGG16 networks trained on synthetic data, then gradual retrained on NAIP data through our gradual unfreezing approach

[43] Yeh, et.al. 2019

would show TCAV results increasingly similar to those networks only trained on NAIP Remote Sensing data. As the VGG16 sets undergoing gradual unfreezing and transfer learning, the TCAV results should become closer to the NAIP trained VGG16 networks. Because our tensor/vectors all contain 88 values, all their values are normalized between 0 and 1. then cosine similarity calculations are useful.

Handling Missing Values in 88 Dimensional Tensors / Vectors

We examined both TensorFlow's cosine similarity implementation and Scikit-learn's cosine similarity implementation. Each had different mechanisms to handle missing data. TensorFlow applies a '0' to the missing value, while Scikit-learn applies the average of the extant value from the other matching fields in the vectors / tensors in our dataset. That is, if the third value was missing from a particular vector / tensor, Scikit would use the average of the third values of the other tensors /vectors in the dataset.

Neither of these approaches were useful because of the variance between the TCAV results of each VGG network. So, we opted to use the few non-significant value results from our TCAV experiments.

Our rationale for this option was that they were likely closer to the expected value for the missing element, than either other options from TensorFlow or Scikit.

This understanding was based on early observation of the TCAV experiments. Specifically, we noticed during earlier TCAV experiments with smaller number of random sets of data to conduct the Welch's t-test, a higher percentage of our results were non-significant as would be expected. When we increased the number of sets and corresponding number of t-tests, which in-turn improved the p-values, the results

produced a higher percentage of significant results, as would be expected. Importantly, the non-significant TCAV result of an earlier test was frequently near the significant TCAV result of the later tests.

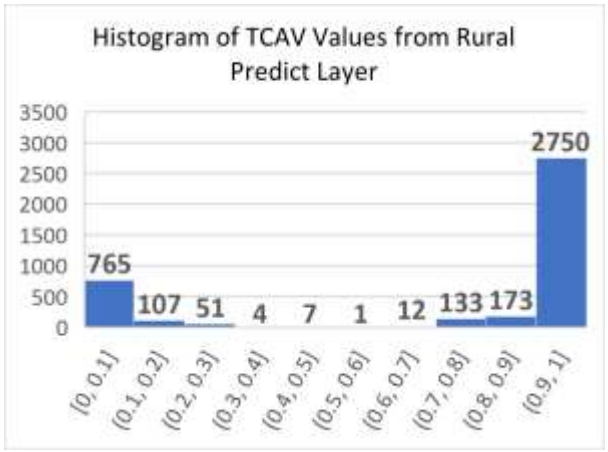
However, to validate this observation and ensure we would not induce too much error in this approach to handle missing data, we examined the error between the TCAV result averages that were statistically significant and TCAV result averages of all values, including the non-significant values. This was done for our 35 VGG16 networks on the first 64 Image Concepts, against the five bottleneck layers.

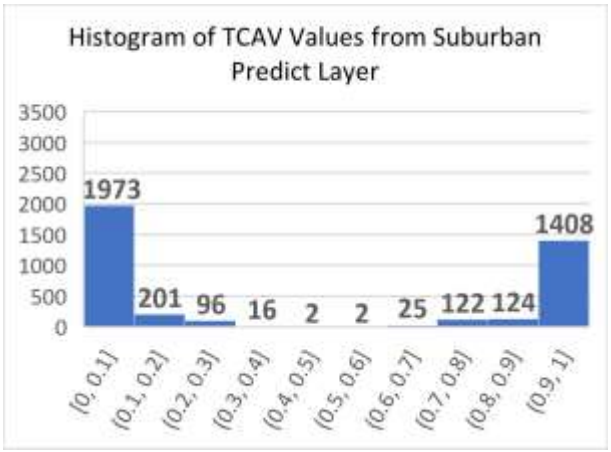
For the Rural class, we found the average of the errors to be 2.18%, median of the errors to be 1.27%, and the maximum error was 15.4% for the Suburban image concept on the 2nd fully connected layer. For the Suburban class, the average of the errors was 1.60%, the median of the errors was 1.24%, and the maximum error was 13.5% for the Rural concept in the 1st fully connected layer. For the Urban class, the average of the errors was 1.61%, the median of the errors was 1.34%, and the maximum error was 13.2% for the pesky Suburban image concept in the 2nd fully connected layer.

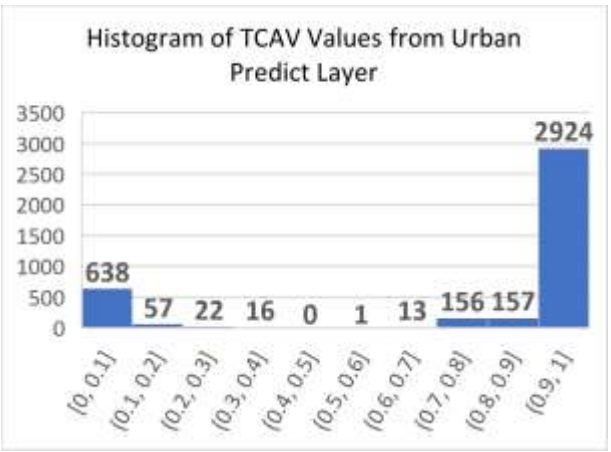
In general, this actually provided some clarity to our TCAV results of the Rural, Suburban, and Urban data we used as image concepts. We intended to use these as sanity checks. For example, in the Urban class, maximum error was for the target concept of Suburban imagery. As you may recall, half of our randomly selected counter-concepts were also suburban imagery, so the T-Test often did not show a distinction.

Bifurcated Values

As can be seen, the TCAV values were bifurcated, mostly close to 1 or to 0. On the right are histograms of the TCAV results of the 35 VGG16 models for the three classes, (Rural, Suburban, and Urban). This bifurcation is primarily driven by Google's design of TCAV in that they set up their metric from 0 to 1 and that a series of random perturbations of would average to 0.5. This resulted in the majority of concept TCAV values that were close to 0.5 of being inside the margin of error (+/-) of the random counter concept perturbations. As such, they would not be considered statistically significant, and would be excluded from further calculations. Because of this bifurcation, we use the Mode, in addition to median and average as one of the sets of values to which the transform learning experiments would be measured.







RESULTS AND DISCUSSION

Discussion of TCAV of 35 VGG16s Trained on NAIP Imagery

To establish our baseline, for each of the three (3) classes of Rural, Suburban, and Urban, we used 35 VGG16 Networks trained only on NAIP data. Using 64 concepts, across each class, we assessed the concepts' perturbations at five layers, Block 4 Max Pool, Block 5 Max Pool, Fully Connected 1, Fully Connected 2, and the Predict Layer. The three graphics below are summary statistics of the TCAV values, Mean, median and mode for all 35 VGG16 Networks trained only on NAIP data.

For these summary statistics, we only used the TCAV T-Test results that were statistically significant, under a p value of 0.05. A review of the TCAV code confirmed that the authors used SciPy to process the T-Tests, correctly handling two-sided T-Tests and splitting the p value for each side.

Notice the Standard Deviation of the TCAV values of the last five VGG16 layers. As can be seen, the Predict layers showed the least variance. Traversing down from the Block 4 Pool to Predict, for two of the classes, Rural and Urban a trend of improving Standard Deviation was exceedingly clear. While the Suburban's Predict layer had the lowest variance for some of the concepts, it's Block 5 Max Pool showed the most consistent values overall.

Specifically, the average of the σ of each 64 concepts across the Rural class was in 0.42, 0.35, 0.32, 0.27, 0.24 for the Block 4 Max Pool, Block 5 Max Pool, Fully Connected 1, Fully Connected 2, and the Predict Layer, respectfully. Likewise, the average Suburban class σ was 0.32, 0.28, 0.31, 0.31, and 0.33 in the same order and the average Urban class σ was 0.41, 0.34, 0.33, 0.30, 0.27.

Because of the lower average σ for the Rural and Urban classes' Predict layer, we selected to use the TCAV values only from the Predict layers for the Gradual Unfreezing Experiments.

Additionally, we selected to add an additional 24 concepts, and use only Predict layer TCAV values for those new concepts. We are presenting them as continuous values, but for the eagle eyed, we present values as a courtesy.

. In APPENDIX 3 TCAV 35 NAIP VGG16 RESULT HEATMAPSwe show our tests of 88 different Image Concepts against 35 editions of VGG16 models trained from scratch on the NAIP data. Because we noticed varied responses for Image Concepts that included strong sensitivity for certain classes, but not for all editions of the VGG16 models, we are using Mean, Median, and Mode for our target TCAV values, and as noted only for the Predict layer in our Gradual Unfreezing Experiments.

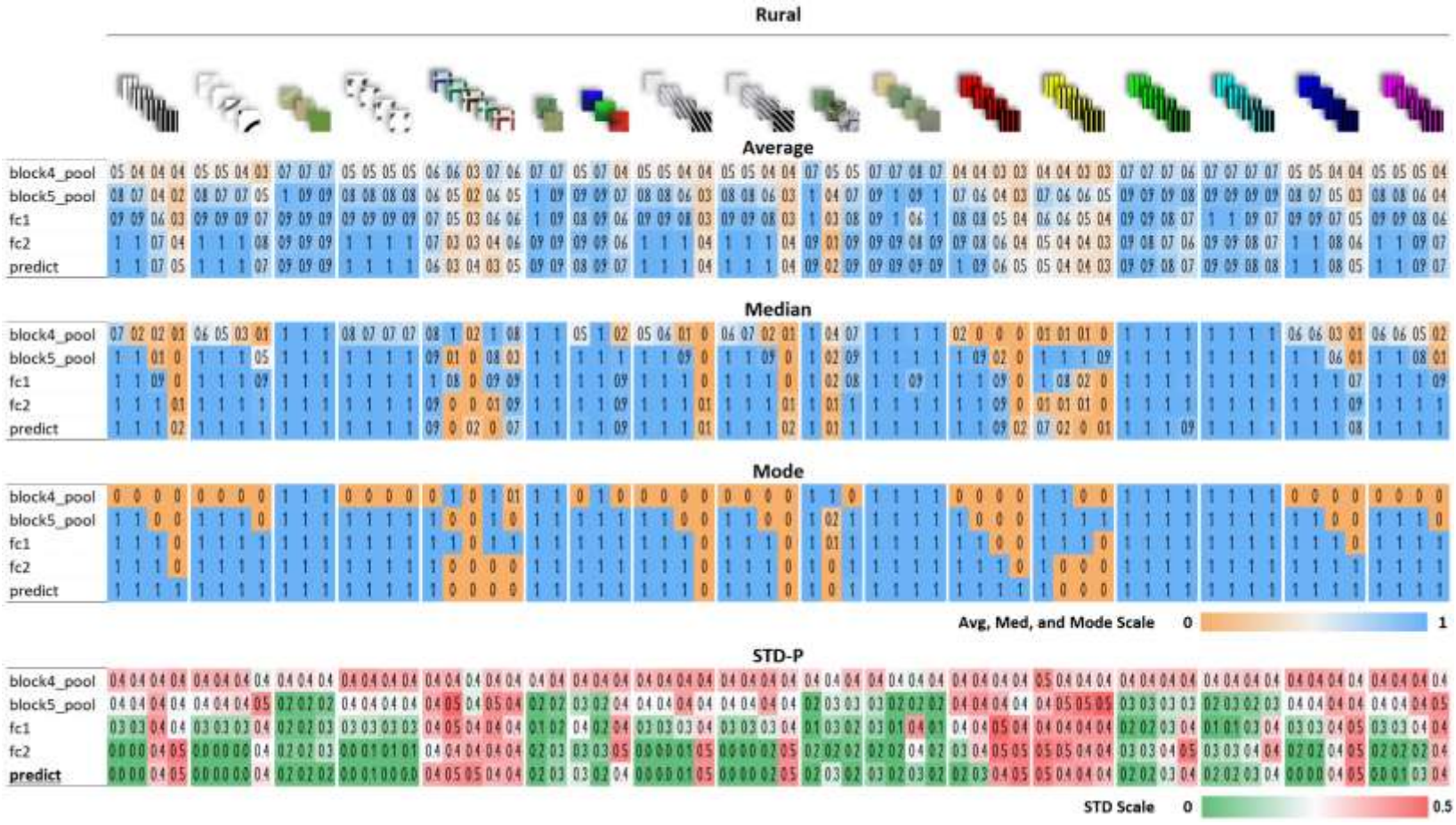


Figure 21 Summary Statistics for Significant TCAV results for 64 Image Concepts in the Rural Class

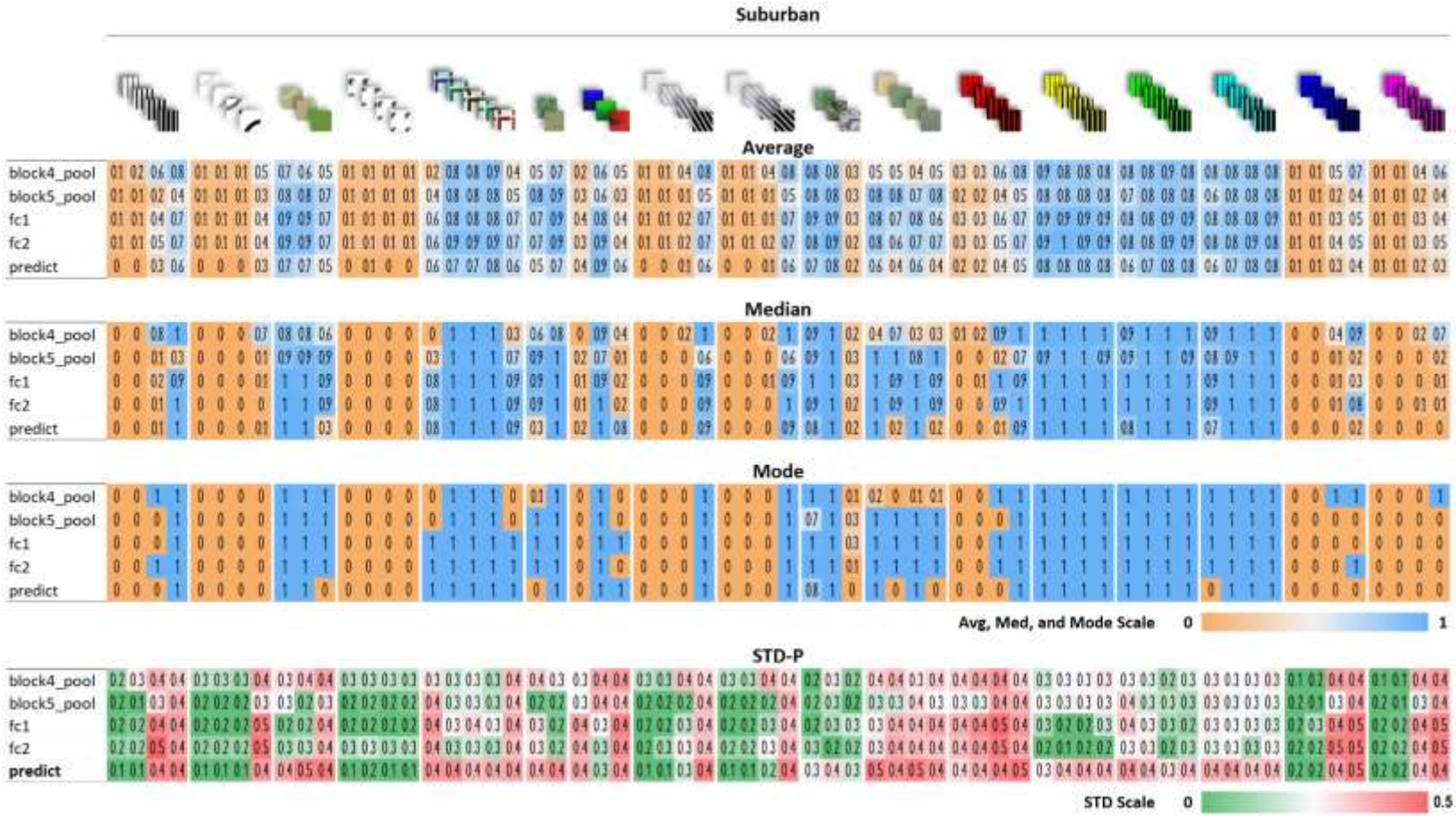


Figure 22 Summary Statistics for Significant TCAV results for 64 Image Concepts in the Suburban Class

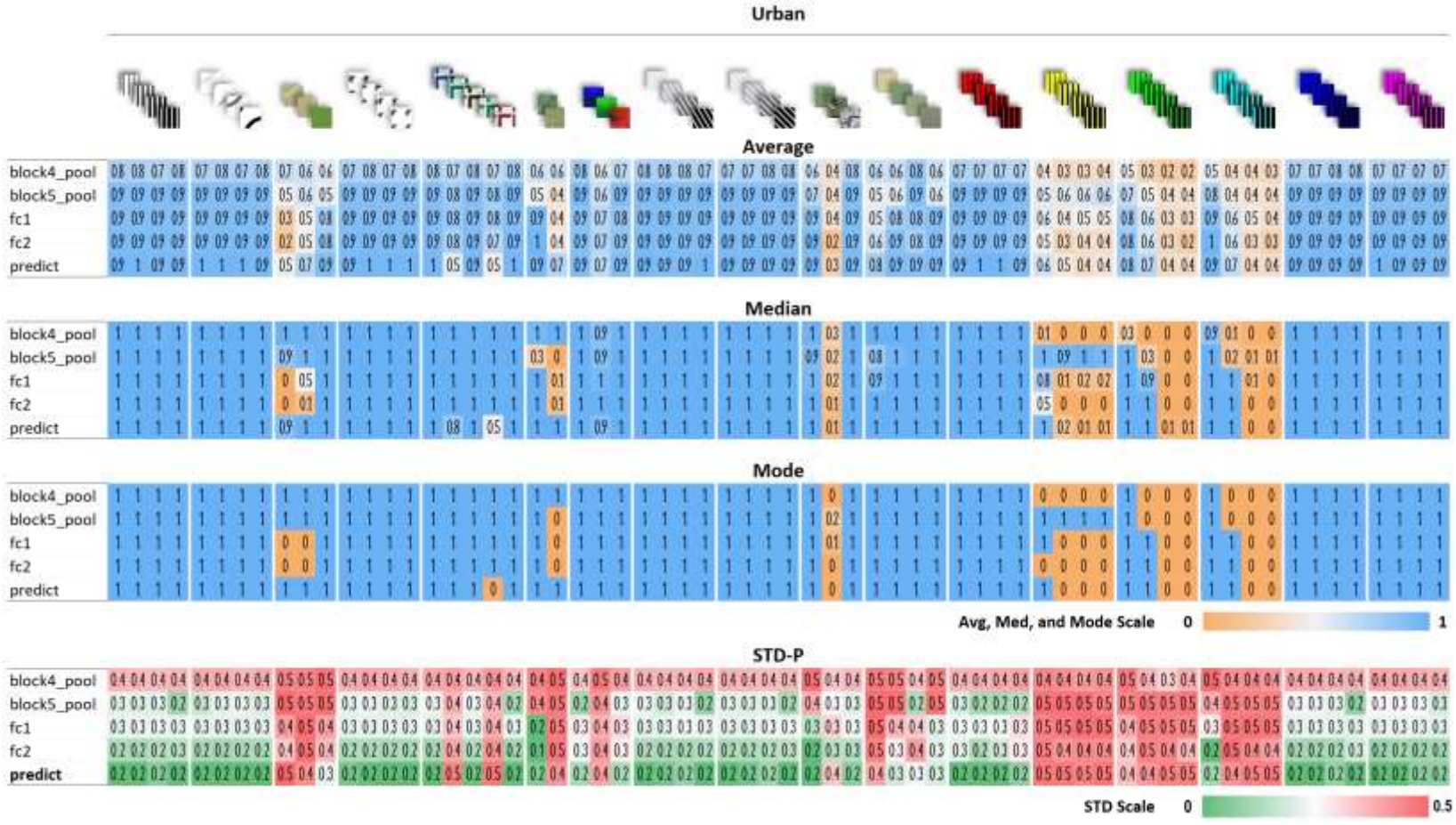


Figure 23 Summary Statistics for Significant TCAV results for 64 Image Concepts in the Urban Class

Transfer Learning Experiments

These experiments were to take a cursory examination the usefulness of TCAV to measure changes induced by Transfer Learning. These initial experiments followed the very standard approach of keeping the lower VGG16 layers frozen. To match typical approaches, the weights from the convolutional layers were transferred from pre-training on synthetic datasets. Also, Block 5 and decision block were unfrozen, while Blocks 1 through 4 were frozen. The classification block was initiated at the default Glorot distribution. We used a drop out of 0.5 between the two fully connected layers in all of the VGG16 networks, save the second network trained on UC Merced Land Use data.

Results

The most striking result is the VGG16 pretrained on ImageNet, then finetuned on NAIP data, had nearly perfect accuracy of the NAIP test set, while it was not sensitive to any of the TCAV Image Concepts, for any class. Immediately, we can see that accuracy and TCAV sensitivity are not directly related. The networks based on the three synthetic datasets and the two land use datasets appeared more sensitive to the learned image concepts of aligned to the targets of the NAIP data.

Interestingly, the grayscale version of the one of the synthetic datasets, Shape & Letter, Desaturated", produced relatively strong accuracy on the target dataset after the round of typical fine tuning. Nonetheless, all the interesting variances in the results can be merely from the Glorot random initialization. As such, we will wait on conducting similarity measures until we conduct the more controlled Gradual Unfreezing experiments. the results can be distances will be conducted in the Gradual

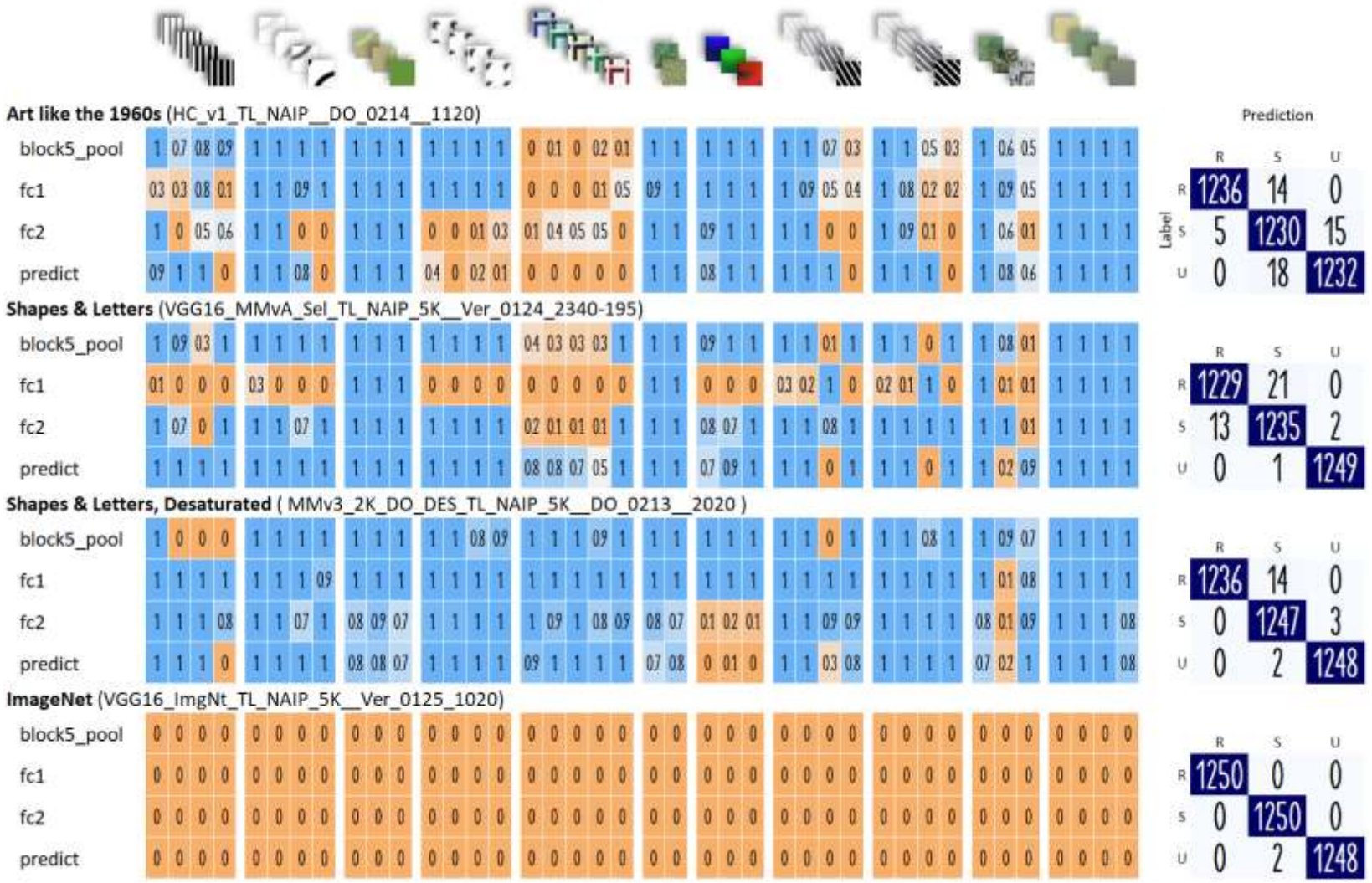


Figure 24 TCAV Rural Class: Results for synthetic models Fine Tuned on NAIP Data 1 of 2

Figure 25 Rural Class: Results for synthetic models Fine Tuned on NAIP Data 2 of 2

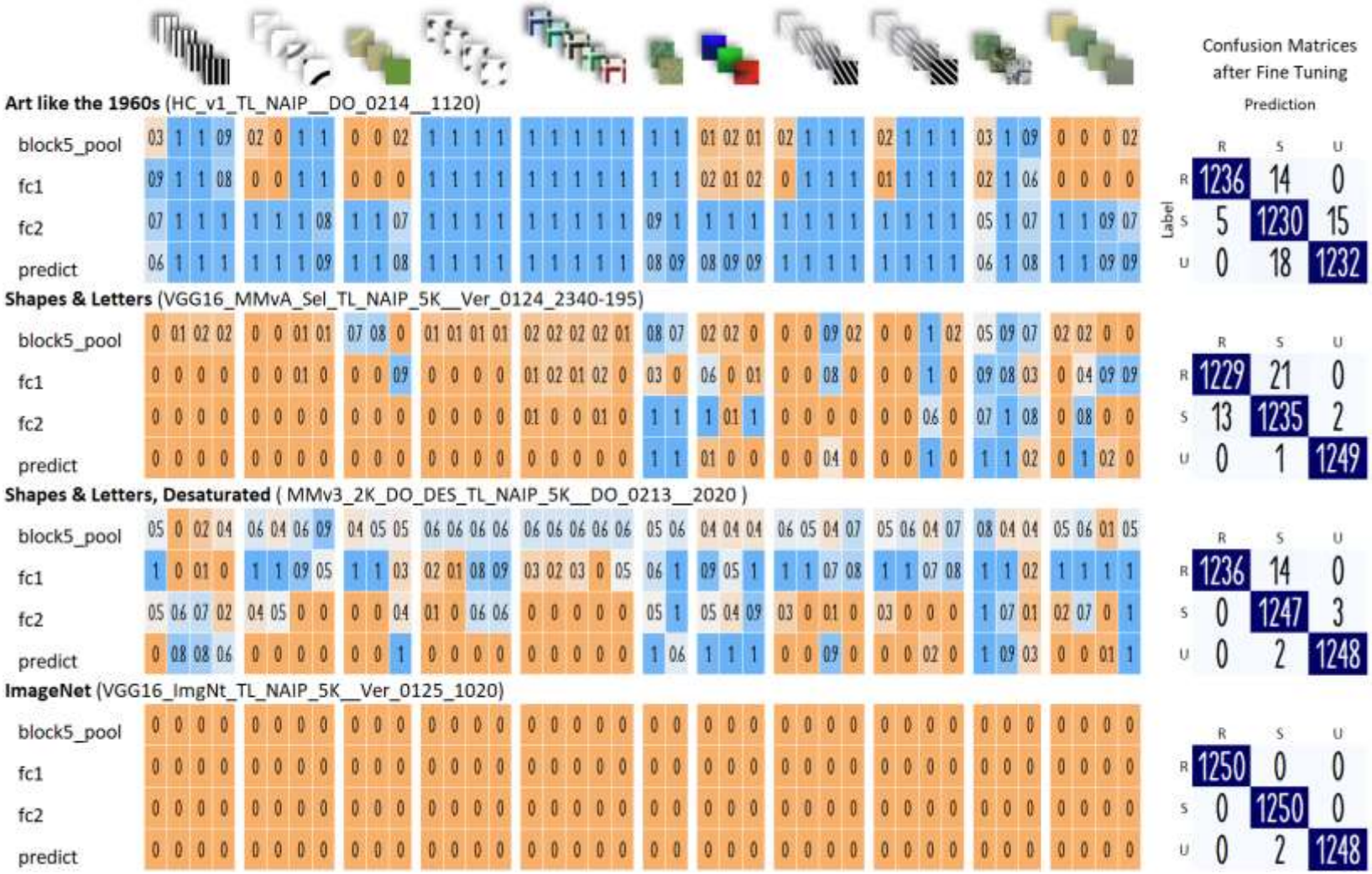


Figure 26 Suburban Class TCAV Results for synthetic models Fine Tuned on NAIP Data 1 of 2

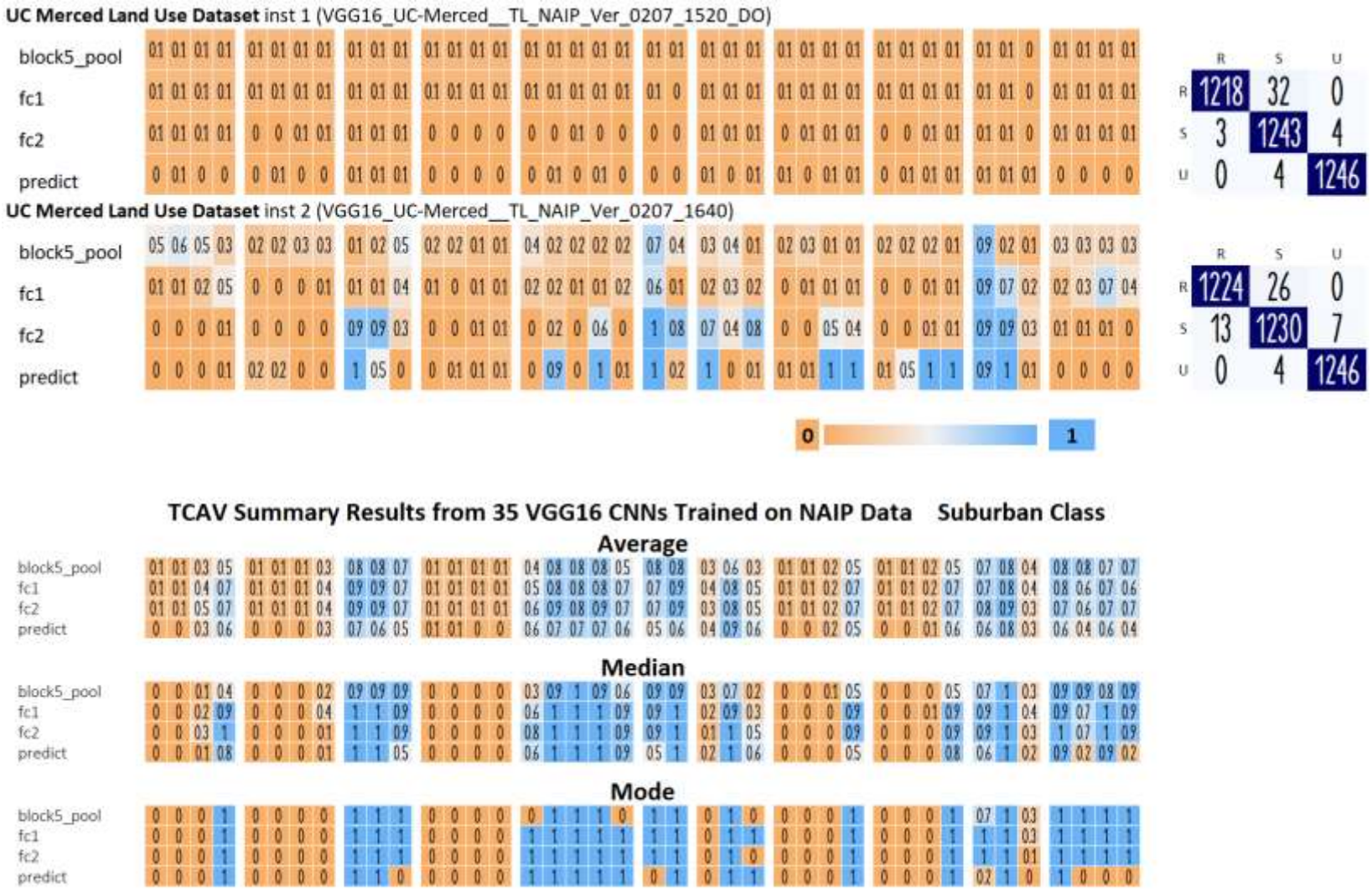


Figure 27 Suburban Class TCAV Results for synthetic models Fine Tuned on NAIP Data 2 of 2

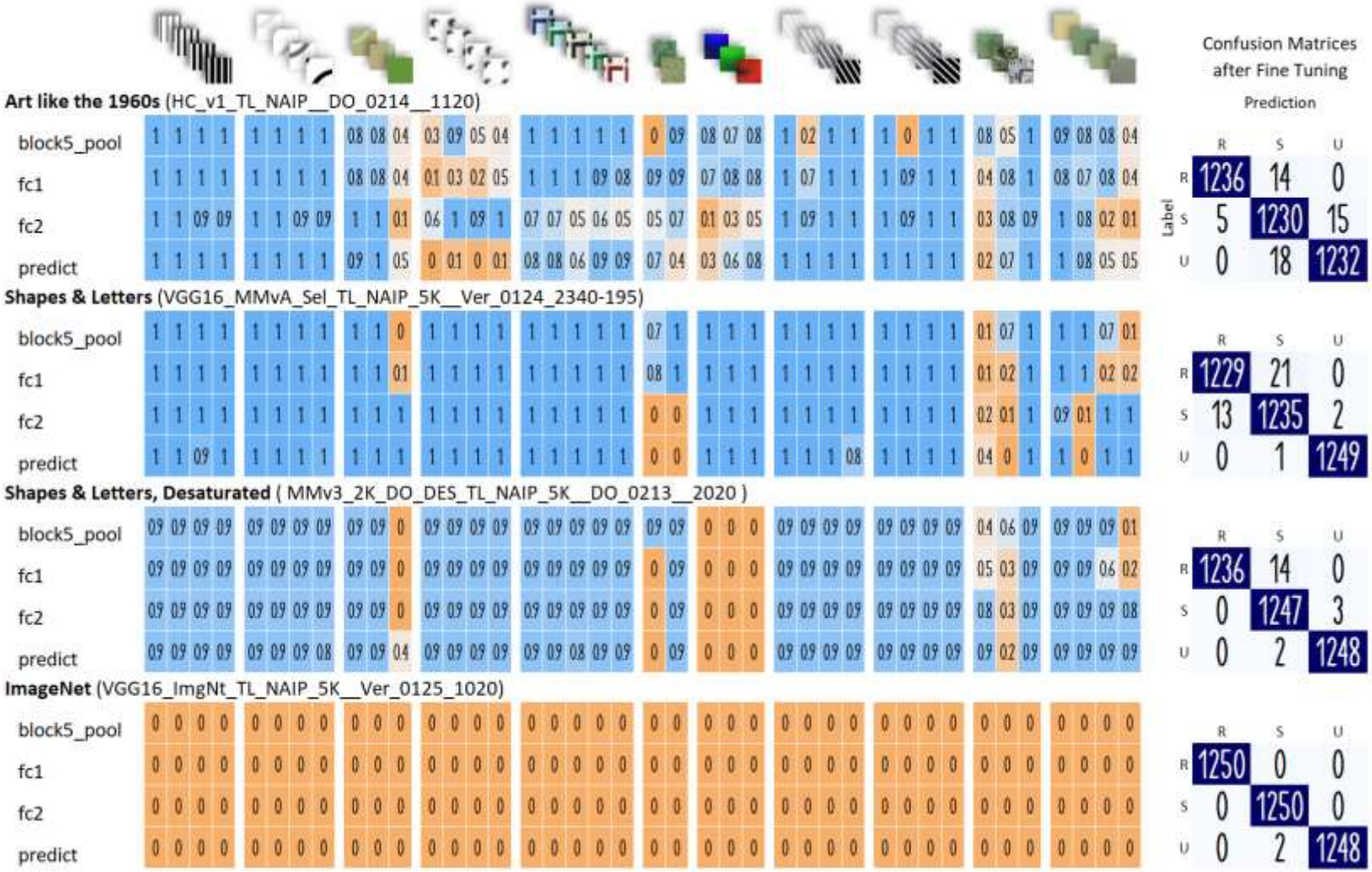


Figure 28 Urban Class TCAV Results for synthetic models Fine Tuned on NAIP Data 1 of 2

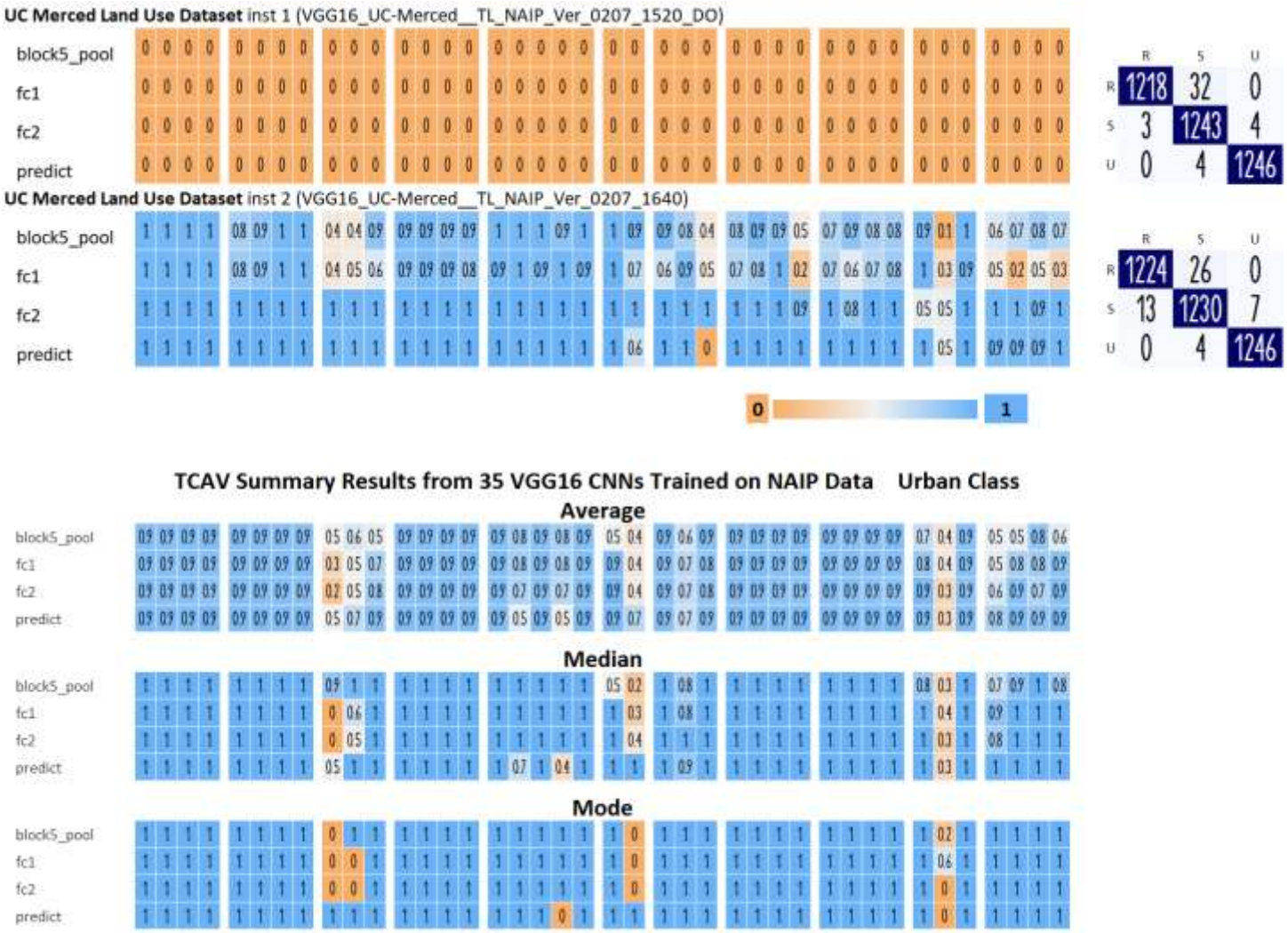


Figure 29 Urban Class TCAV Results for synthetic models Fine Tuned on NAIP Data 2 of 2

Gradual Unfreezing - Are TCAV Values reflective of Accuracy?

The two sets of five VGG16 networks were trained on synthetic data that was built similar to some features of the Rural, Suburban, and Urban imagery from NAIP Remote Sensing data. The first set, identified as Version 1 had relatively primitive imagery features for Rural, Suburban, and Urban. The second set, Version 2, used the same Rural and Suburban synthetic imagery, but the Urban imagery was more reflective of the features of the Urban class. …

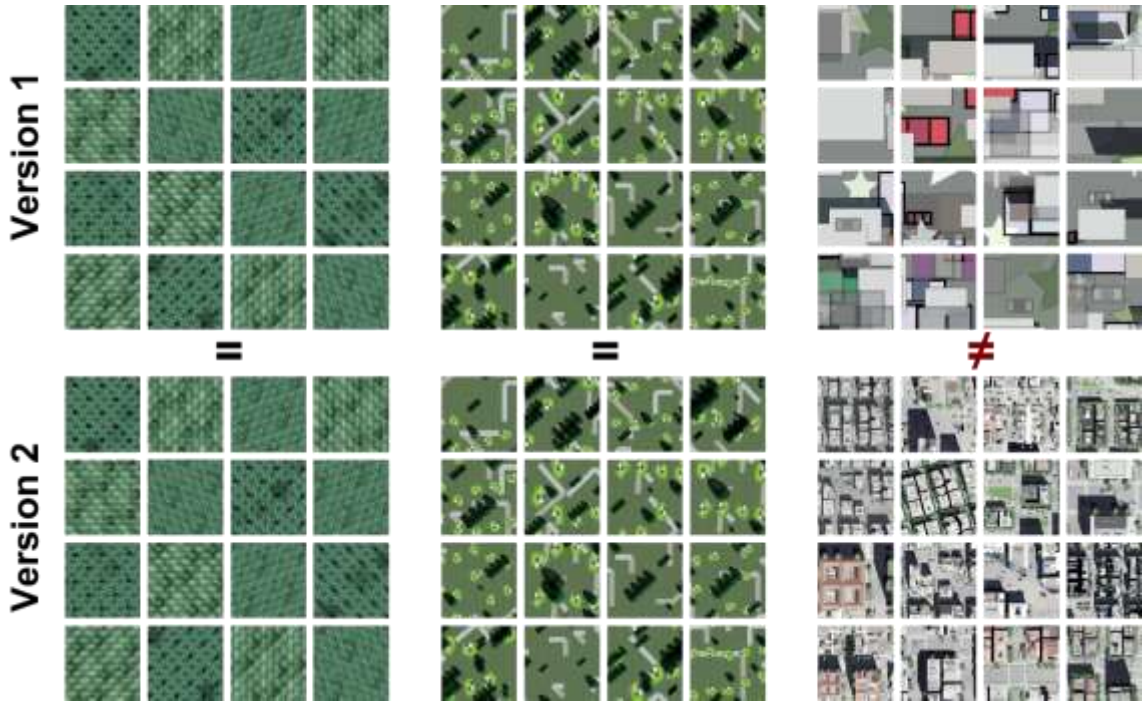


Figure 30 Two versions of Synthetic Rural, Suburban, and Urban imagery

As each set transverses the gradual unfreezing and transfer learning process, the TCAV values and the corresponding accuracy were recorded at each step. That is, the values of both TCAV and accuracy were recorded at each stage of the gradual unfreezing

process starting with recording the values of the synthetic data trained VGG16 networks, then recording the value of after the networks experienced transfer learning of only the fully connected layers, then after the next iteration of transfer learning of fully connected layers and Block 5, and so forth.

Results

Although TCAV values changed during our Gradual Unfreezing Experiments, the results clearly show they were not commensurate with the changes in accuracy values, *for our use*. That is, our intent is to develop imagery that could be used as a good source domain for transfer learning such that it is reflective of the target domain.

In the examples shown on the following pages, we show the confusion matrices of a set of VGG16 networks that were trained on synthetic data, but slowly retrained Remote Sensing data through our gradual unfreezing process.

Similar to the initial Transfer Learning experiment, the results of the Gradual Unfreezing experiment show that accuracy and learned concepts are not necessarily related. Most of our Kendal Tau results are not statistically significant, the couple that are significant show negative correlation and that increased accuracy slightly correlates to further distance from our target TCAV values. There are a few elements that may contribute to this outcome. We selected to use only Predict layer values because of their tendency to be more consistent. As such, once the first iteration of fine tuning on the classification block alone, there may be only incremental changes to this layer as the other layers are unfrozen. Some results below and in APPENDIX 2 TCAV RESULTS FROM GRADUAL UNFREEZING align with this concept. The other element that may

contribute to this is revealed in the original TCAV results to establish a baseline. That is, the learned concepts, even in the five instances of each set of synthetic data did not show uniformity in the few concepts that were present.

This was consistent with our early evaluations of VGG16 networks trained from scratch on NAIP data. Some showed completely opposite results from the rest of the cohort of 35. See APPENDIX 3 TCAV 35 NAIP VGG16 RESULT HEATMAPS

Lastly, but importantly, lack of strong correlation between changes of learned concepts and transfer learning, aligns with Raghu et al[44] findings of internal network representations, where they report that transfer learning does not strongly influence the changes of deep networks and the network representations. Large networks, like VGG16 have sufficient trainable parameters, that they can train parameters to achieve accuracy targets and minimize loss, but those parameters may not directly impact other parameters of learned concepts. In essence, an abundance of neurons allows them to learn multiple tasks without too much penalty.

Conclusion

So, for our use, using TCAV would not feasibility produce useful values to guide practitioners in the development of source synthetic datasets for large networks like VGG16. This is also reinforced by the TCAV values of the 35 networks trained only on NAIP data. Using TCAV as a distance measure might work for smaller networks, but that could obviate the need for transfer learning. So, in summary, we might find alternatives to TCAV that might fare better for this use.

[44] Raghu, et.al 2019.

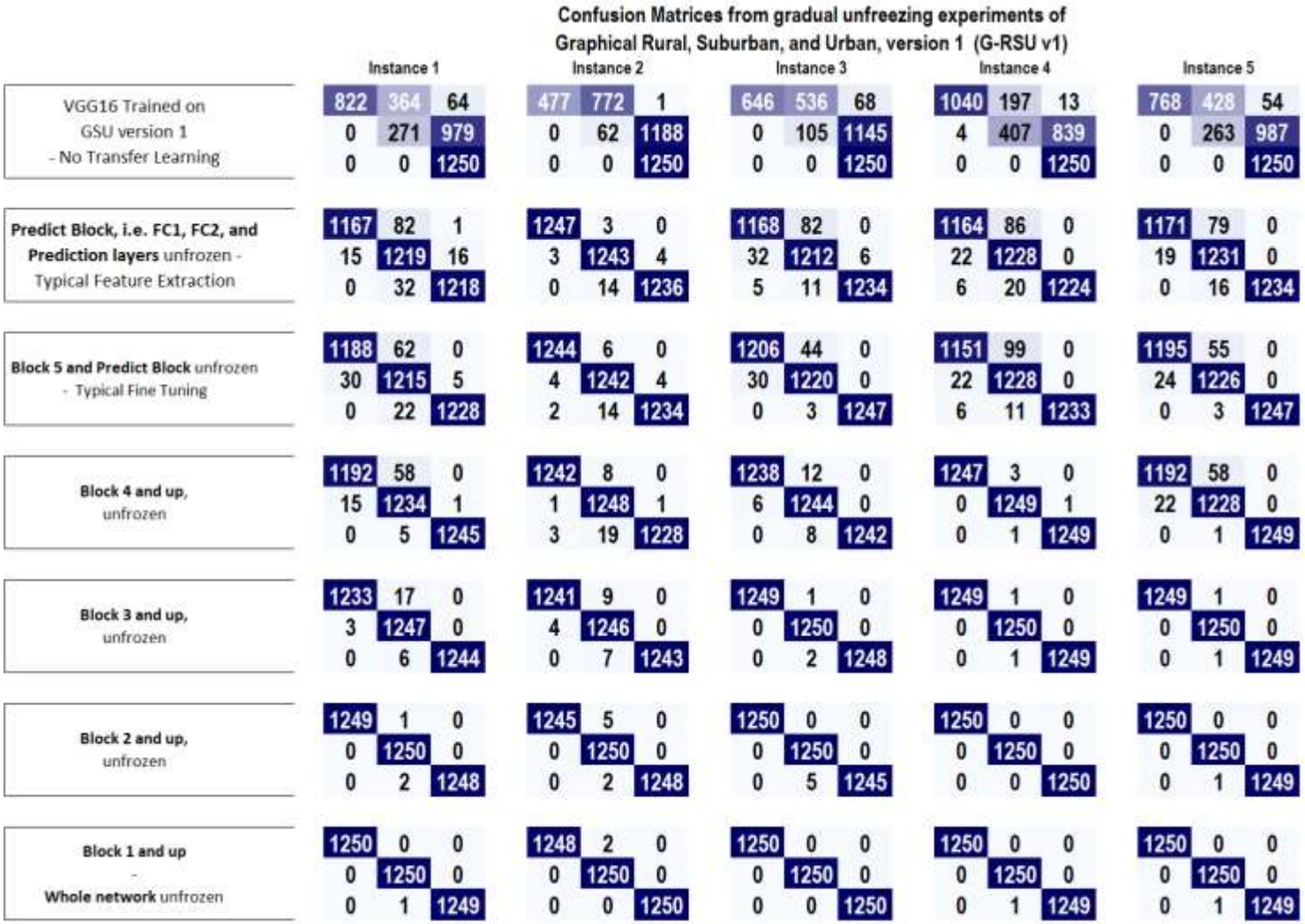


Figure 31 Confusion Matrices from Gradual Unfreezing Experiments of G-RSU v1

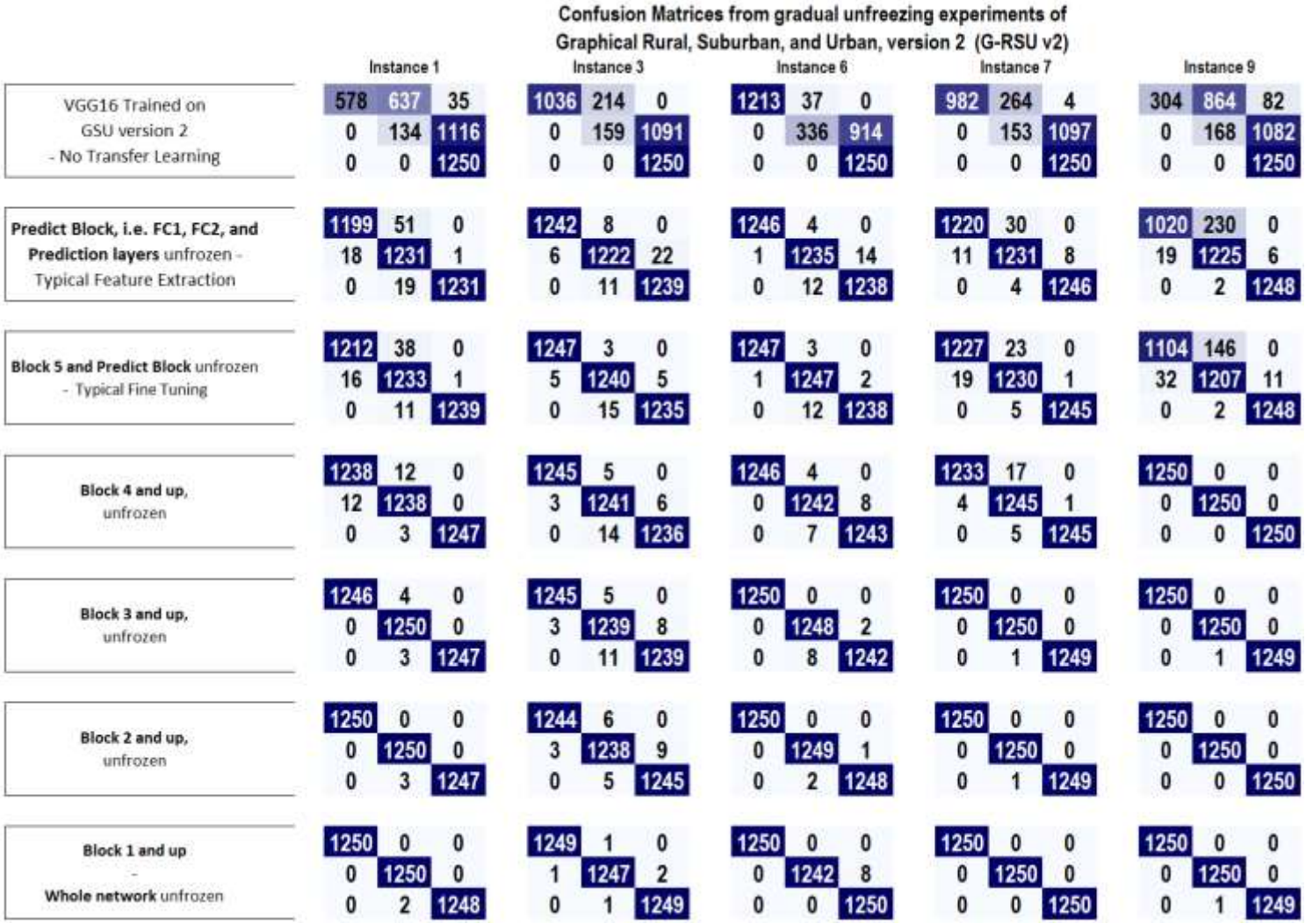
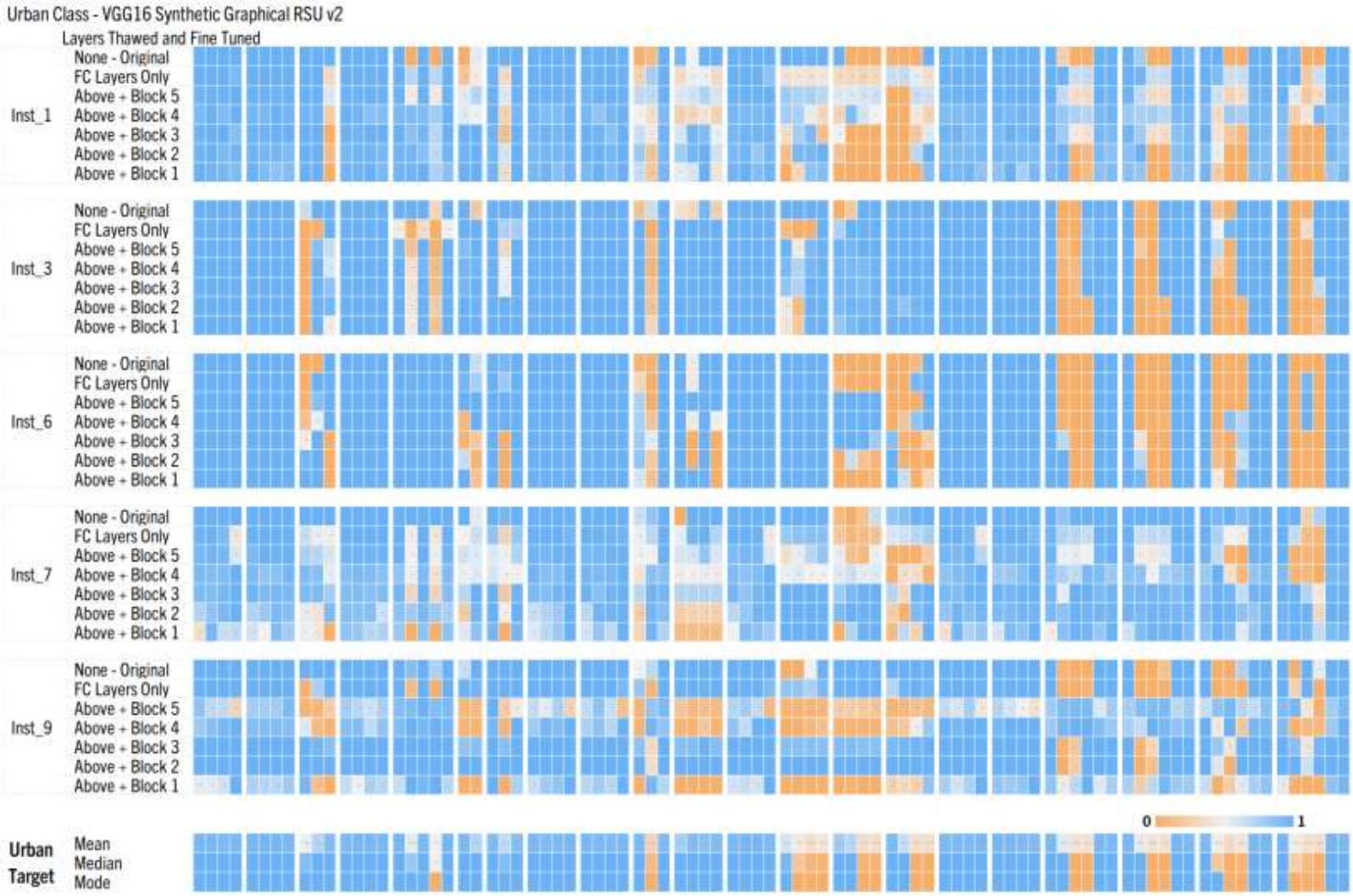


Figure 32 Confusion Matrices from Gradual Unfreezing Experiments of G-RSU v2

Please Note: Remainder of Gradual Unfreezing Tables are in the Appendix 2

Figure 33 Five Gradual Unfreezing Experiments using G-RSU v1 Synth Dataset, Urban Class



Please Note: Remainder of Gradual Unfreezing Tables are in the Appendix 2

Figure 34 Five Gradual Unfreezing Experiments using G-RSU v2 Synth Dataset, Urban Class

Table 1 Summary Results of Gradual Unfreezing Experiments for G-RSU v1

Graphical Rural, Suburban, Urban **v1**

	Overall Accuracy	Rural Recall	Suburban Recall	Urban Recall	Rural Class TCAV Cosine Similarity to: NAIP Mean	Rural Class TCAV Cosine Similarity to: NAIP Median	Rural Class TCAV Cosine Similarity to: NAIP Mode	Suburban Class TCAV Cosine Similarity to: NAIP Mean	Suburban Class TCAV Cosine Similarity to: NAIP Median	Suburban Class TCAV Cosine Similarity to: NAIP Mode	Urban Class TCAV Cosine Similarity to: NAIP Mean	Urban Class TCAV Cosine Similarity to: NAIP Median	Urban Class TCAV Cosine Similarity to: NAIP Mode
G_RSU_v1___Inst_1___Original	0.625	0.658	0.217	1.000	0.96	0.95	0.95	0.53	0.51	0.49	0.95	0.88	0.87
G_RSU_v1___Inst_1___Pred___Block	0.961	0.934	0.975	0.974	0.95	0.93	0.93	0.80	0.79	0.74	0.96	0.92	0.92
G_RSU_v1___Inst_1___Block_5_Above	0.968	0.950	0.972	0.982	0.96	0.95	0.93	0.81	0.79	0.74	0.97	0.94	0.93
G_RSU_v1___Inst_1___Block_4_Above	0.979	0.954	0.987	0.996	0.96	0.97	0.94	0.84	0.79	0.75	0.97	0.93	0.92
G_RSU_v1___Inst_1___Block_3_Above	0.993	0.986	0.998	0.995	0.96	0.97	0.94	0.85	0.78	0.74	0.93	0.88	0.87
G_RSU_v1___Inst_1___Block_2_Above	0.999	0.999	1.000	0.998	0.95	0.94	0.92	0.87	0.80	0.76	0.94	0.89	0.88
G_RSU_v1___Inst_1___Block_1_Above	1.000	1.000	1.000	0.999	0.94	0.94	0.91	0.91	0.85	0.80	0.94	0.90	0.89
G_RSU_v1___Inst_2___Original	0.477	0.382	0.050	1.000	0.96	0.93	0.92	0.86	0.85	0.81	0.00	0.00	0.00
G_RSU_v1___Inst_2___Pred___Block	0.994	0.998	0.994	0.989	0.97	0.95	0.95	0.88	0.82	0.81	0.96	0.91	0.90
G_RSU_v1___Inst_2___Block_5_Above	0.992	0.995	0.994	0.987	0.97	0.95	0.95	0.85	0.78	0.76	0.89	0.80	0.80
G_RSU_v1___Inst_2___Block_4_Above	0.991	0.994	0.998	0.982	0.96	0.97	0.94	0.84	0.79	0.75	0.97	0.93	0.92
G_RSU_v1___Inst_2___Block_3_Above	0.995	0.993	0.997	0.994	0.96	0.95	0.95	0.72	0.62	0.60	0.88	0.82	0.81
G_RSU_v1___Inst_2___Block_2_Above	0.998	0.996	1.000	0.998	0.91	0.89	0.89	0.76	0.65	0.63	0.89	0.82	0.82
G_RSU_v1___Inst_2___Block_1_Above	0.999	0.998	1.000	1.000	0.97	0.95	0.95	0.77	0.67	0.64	0.95	0.90	0.89
G_RSU_v1___Inst_3___Original	0.534	0.517	0.084	1.000	0.94	0.93	0.90	0.70	0.64	0.60	0.94	0.91	0.90
G_RSU_v1___Inst_3___Pred___Block	0.964	0.934	0.970	0.987	0.95	0.93	0.91	0.70	0.60	0.56	0.92	0.91	0.90
G_RSU_v1___Inst_3___Block_5_Above	0.979	0.965	0.976	0.998	0.96	0.94	0.93	0.78	0.69	0.66	0.96	0.94	0.92
G_RSU_v1___Inst_3___Block_4_Above	0.993	0.990	0.995	0.994	0.95	0.92	0.91	0.91	0.84	0.81	0.95	0.92	0.91
G_RSU_v1___Inst_3___Block_3_Above	0.999	0.999	1.000	0.998	0.95	0.92	0.91	0.76	0.65	0.63	0.96	0.92	0.90
G_RSU_v1___Inst_3___Block_2_Above	0.999	1.000	1.000	0.996	0.93	0.91	0.90	0.86	0.81	0.77	0.96	0.91	0.89
G_RSU_v1___Inst_3___Block_1_Above	1.000	1.000	1.000	1.000	0.93	0.90	0.88	0.77	0.67	0.65	0.90	0.86	0.85
G_RSU_v1___Inst_4___Original	0.719	0.832	0.326	1.000	0.95	0.92	0.91	0.64	0.59	0.58	0.97	0.92	0.91
G_RSU_v1___Inst_4___Pred___Block	0.964	0.931	0.982	0.979	0.97	0.97	0.94	0.69	0.61	0.59	0.91	0.88	0.87
G_RSU_v1___Inst_4___Block_5_Above	0.963	0.921	0.982	0.986	0.97	0.96	0.95	0.70	0.61	0.59	0.91	0.89	0.88
G_RSU_v1___Inst_4___Block_4_Above	0.999	0.998	0.999	0.999	0.93	0.92	0.90	0.65	0.53	0.50	0.94	0.89	0.89
G_RSU_v1___Inst_4___Block_3_Above	0.999	0.999	1.000	0.999	0.94	0.93	0.90	0.68	0.58	0.56	0.94	0.89	0.88
G_RSU_v1___Inst_4___Block_2_Above	1.000	1.000	1.000	1.000	0.94	0.93	0.90	0.77	0.69	0.66	0.93	0.88	0.87
G_RSU_v1___Inst_4___Block_1_Above	1.000	1.000	1.000	0.999	0.94	0.93	0.90	0.73	0.66	0.63	0.92	0.86	0.85
G_RSU_v1___Inst_5___Original	0.608	0.614	0.210	1.000	0.94	0.92	0.89	0.79	0.75	0.71	0.94	0.93	0.91
G_RSU_v1___Inst_5___Pred___Block	0.970	0.937	0.985	0.987	0.96	0.95	0.93	0.74	0.64	0.61	0.92	0.91	0.89
G_RSU_v1___Inst_5___Block_5_Above	0.978	0.956	0.981	0.998	0.92	0.91	0.88	0.77	0.68	0.65	0.95	0.94	0.93
G_RSU_v1___Inst_5___Block_4_Above	0.978	0.954	0.982	0.999	0.95	0.93	0.90	0.81	0.72	0.68	0.94	0.93	0.92
G_RSU_v1___Inst_5___Block_3_Above	0.999	0.999	1.000	0.999	0.97	0.96	0.94	0.85	0.76	0.73	0.96	0.90	0.89
G_RSU_v1___Inst_5___Block_2_Above	1.000	1.000	1.000	0.999	0.97	0.96	0.94	0.87	0.80	0.76	0.96	0.90	0.90
G_RSU_v1___Inst_5___Block_1_Above	1.000	1.000	1.000	0.999	0.97	0.96	0.93	0.86	0.79	0.75	0.96	0.91	0.90

Table 2 Summary Results of Gradual Unfreezing Experiments for G-RSU v2

Graphical Rural, Suburban, Urban **v2**

	Overall Accuracy	Rural Recall	Suburban Recall	Urban Recall	Rural Class TCAV Cosine Similarity to: NAIP Mean	Rural Class TCAV Cosine Similarity to: NAIP Median	Rural Class TCAV Cosine Similarity to: NAIP Mode	Suburban Class TCAV Cosine Similarity to: NAIP Mean	Suburban Class TCAV Cosine Similarity to: NAIP Median	Suburban Class TCAV Cosine Similarity to: NAIP Mode	Urban Class TCAV Cosine Similarity to: NAIP Mean	Urban Class TCAV Cosine Similarity to: NAIP Median	Urban Class TCAV Cosine Similarity to: NAIP Mode
G_RSU_v2___Inst_1___Original	0.523	0.462	0.107	1.000	0.95	0.93	0.92	0.71	0.70	0.66	0.93	0.91	0.91
G_RSU_v2___Inst_1___Pred___Block	0.976	0.959	0.985	0.985	0.90	0.88	0.87	0.81	0.77	0.73	0.96	0.92	0.91
G_RSU_v2___Inst_1___Block_5_Above	0.982	0.970	0.986	0.991	0.96	0.95	0.91	0.76	0.71	0.67	0.97	0.94	0.93
G_RSU_v2___Inst_1___Block_4_Above	0.993	0.990	0.990	0.998	0.98	0.97	0.95	0.80	0.79	0.74	0.95	0.91	0.89
G_RSU_v2___Inst_1___Block_3_Above	0.998	0.997	1.000	0.998	0.97	0.98	0.96	0.63	0.61	0.57	0.95	0.94	0.93
G_RSU_v2___Inst_1___Block_2_Above	0.999	1.000	1.000	0.998	0.97	0.98	0.96	0.51	0.47	0.43	0.94	0.93	0.92
G_RSU_v2___Inst_1___Block_1_Above	0.999	1.000	1.000	0.998	0.97	0.98	0.97	0.61	0.58	0.54	0.93	0.93	0.92
G_RSU_v2___Inst_3___Original	0.652	0.829	0.127	1.000	0.91	0.89	0.88	0.79	0.79	0.77	0.93	0.88	0.87
G_RSU_v2___Inst_3___Pred___Block	0.987	0.994	0.978	0.991	0.95	0.93	0.93	0.80	0.77	0.73	0.94	0.88	0.88
G_RSU_v2___Inst_3___Block_5_Above	0.993	0.998	0.992	0.988	0.80	0.77	0.77	0.83	0.82	0.79	0.96	0.91	0.90
G_RSU_v2___Inst_3___Block_4_Above	0.993	0.996	0.993	0.989	0.95	0.93	0.92	0.88	0.82	0.80	0.96	0.91	0.90
G_RSU_v2___Inst_3___Block_3_Above	0.993	0.996	0.991	0.991	0.94	0.93	0.93	0.87	0.82	0.79	0.96	0.91	0.90
G_RSU_v2___Inst_3___Block_2_Above	0.994	0.995	0.990	0.996	0.95	0.92	0.93	0.86	0.81	0.78	0.96	0.94	0.92
G_RSU_v2___Inst_3___Block_1_Above	0.999	0.999	0.998	0.999	0.94	0.93	0.93	0.87	0.82	0.79	0.96	0.93	0.92
G_RSU_v2___Inst_6___Original	0.746	0.970	0.269	1.000	0.92	0.89	0.88	0.88	0.83	0.80	0.93	0.90	0.89
G_RSU_v2___Inst_6___Pred___Block	0.992	0.997	0.988	0.990	0.70	0.69	0.69	0.75	0.68	0.66	0.93	0.90	0.89
G_RSU_v2___Inst_6___Block_5_Above	0.995	0.998	0.998	0.990	0.89	0.86	0.85	0.68	0.62	0.60	0.95	0.92	0.90
G_RSU_v2___Inst_6___Block_4_Above	0.995	0.997	0.994	0.994	0.91	0.89	0.89	0.74	0.64	0.62	0.94	0.90	0.88
G_RSU_v2___Inst_6___Block_3_Above	0.997	1.000	0.998	0.994	0.91	0.88	0.87	0.78	0.69	0.67	0.93	0.91	0.89
G_RSU_v2___Inst_6___Block_2_Above	0.999	1.000	0.999	0.998	0.92	0.89	0.88	0.80	0.73	0.70	0.93	0.92	0.91
G_RSU_v2___Inst_6___Block_1_Above	0.998	1.000	0.994	1.000	0.92	0.89	0.88	0.78	0.69	0.66	0.94	0.93	0.92
G_RSU_v2___Inst_7___Original	0.636	0.786	0.122	1.000	0.97	0.95	0.93	0.88	0.89	0.87	0.95	0.89	0.88
G_RSU_v2___Inst_7___Pred___Block	0.986	0.976	0.985	0.997	0.96	0.95	0.94	0.88	0.84	0.81	0.96	0.92	0.91
G_RSU_v2___Inst_7___Block_5_Above	0.987	0.982	0.984	0.996	0.96	0.95	0.95	0.85	0.80	0.77	0.96	0.93	0.93
G_RSU_v2___Inst_7___Block_4_Above	0.993	0.986	0.996	0.996	0.98	0.96	0.95	0.82	0.74	0.71	0.96	0.93	0.93
G_RSU_v2___Inst_7___Block_3_Above	1.000	1.000	1.000	0.999	0.97	0.96	0.94	0.83	0.74	0.71	0.95	0.89	0.88
G_RSU_v2___Inst_7___Block_2_Above	1.000	1.000	1.000	0.999	0.97	0.96	0.94	0.82	0.72	0.69	0.91	0.84	0.83
G_RSU_v2___Inst_7___Block_1_Above	1.000	1.000	1.000	1.000	0.97	0.96	0.94	0.83	0.74	0.71	0.87	0.79	0.79
G_RSU_v2___Inst_9___Original	0.459	0.243	0.134	1.000	0.96	0.93	0.91	0.76	0.71	0.67	0.96	0.93	0.91
G_RSU_v2___Inst_9___Pred___Block	0.931	0.816	0.980	0.998	0.86	0.84	0.85	0.66	0.61	0.60	0.96	0.93	0.91
G_RSU_v2___Inst_9___Block_5_Above	0.949	0.883	0.966	0.998	0.86	0.84	0.85	0.78	0.73	0.71	0.85	0.79	0.79
G_RSU_v2___Inst_9___Block_4_Above	1.000	1.000	1.000	1.000	0.96	0.94	0.92	0.81	0.74	0.71	0.88	0.84	0.83
G_RSU_v2___Inst_9___Block_3_Above	1.000	1.000	1.000	0.999	0.95	0.93	0.92	0.62	0.59	0.58	0.97	0.92	0.90
G_RSU_v2___Inst_9___Block_2_Above	1.000	1.000	1.000	1.000	0.97	0.95	0.93	0.77	0.68	0.66	0.96	0.91	0.90
G_RSU_v2___Inst_9___Block_1_Above	1.000	1.000	1.000	0.999	0.97	0.96	0.94	0.56	0.45	0.43	0.86	0.82	0.80

Table 3 Kendall Tau Results from Gradual Unfreezing Experiments

Class	Accuracy Measure	TCAV Cos Similarity to:	Kendall Tau	p value
Rural	Overall Accuracy	NAIP Mean	0.12	0.15
	Overall Accuracy	NAIP Median	0.12	0.14
	Overall Accuracy	NAIP Mode	0.10	0.22
	Rural Recall	NAIP Mean	0.05	0.57
	Rural Recall	NAIP Median	0.06	0.45
	Rural Recall	NAIP Mode	0.04	0.62
Suburban	Overall Accuracy	NAIP Mean	0.04	0.60
	Overall Accuracy	NAIP Median	-0.07	0.43
	Overall Accuracy	NAIP Mode	-0.06	0.45
	Suburban Recall	NAIP Mean	0.01	0.88
	Suburban Recall	NAIP Median	-0.10	0.24
	Suburban Recall	NAIP Mode	-0.10	0.26
Urban	Overall Accuracy	NAIP Mean	-0.06	0.44
	Overall Accuracy	NAIP Median	-0.17	0.04
	Overall Accuracy	NAIP Mode	-0.16	0.05
	Urban Recall	NAIP Mean	-0.13	0.12
	Urban Recall	NAIP Median	-0.16	0.06
	Urban Recall	NAIP Mode	-0.18	0.04

FUTURE STUDIES AND OBSERVATION OF 35 VGG16 MODELS

During review of the accuracy data from the Gradual Unfreezing Experiments, we selected to run inference on the synthetic datasets, using our VGG16 networks only trained on NAIP Remote Sensing data. We were shocked how well strongly they recognized the synthetic data without any retraining of the VGG16 networks. The examples include the best and the worst performing networks, as well as representative samples from the rest of the networks. This drove an interest into a review of the logit values to better understand the surprising results.

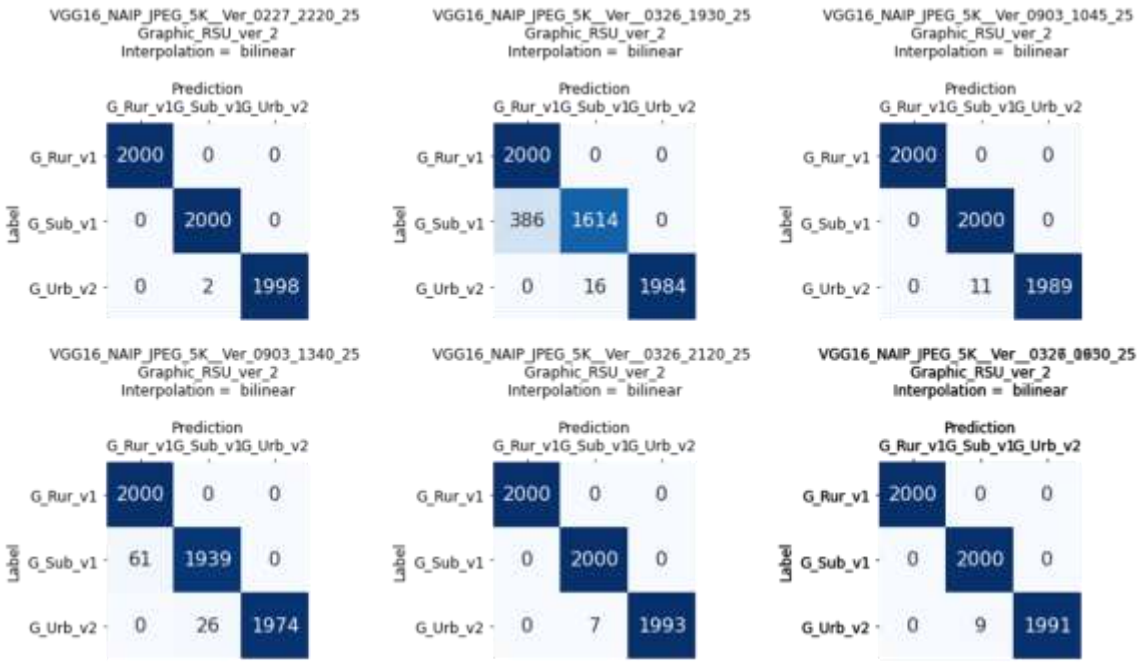


Figure 35 Example Confusion Matrices from NAIP Trained Models and Synth Data

We noticed that the logit values of the Remote Sensing test data and from the synth data followed similar patterns among the 35 networks for two of our classes. That is, when on network had higher predict values compared to another network in the Remote Sensing data, it would follow the same pattern in the synthetic data. This was remarkable consistent for both Urban and Rural.

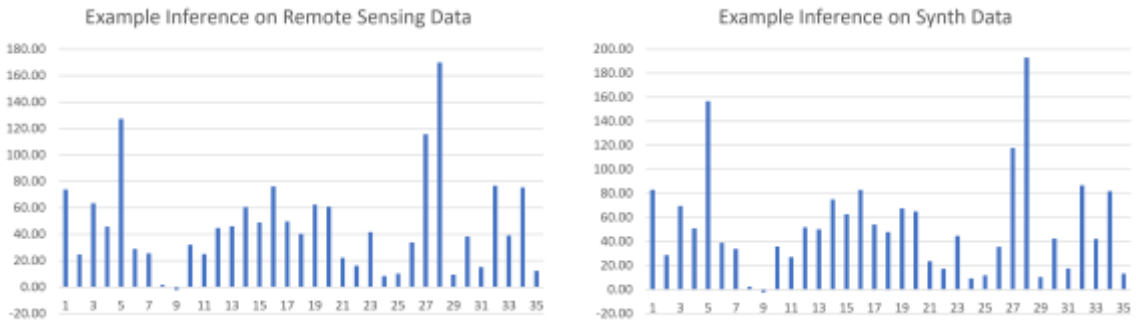


Figure 36 Nearly identical pattern of Logit values for two different data sets.

Table 4 Kendall Tau Results comparing ranks of descriptive statistics for two different datasets

URBAN	Tau	p value
Mean	0.98	9.75E-17
Median	0.99	7.67E-17
Std Dev	0.81	1.03E-11
Min	0.63	8.61E-08
Max	0.95	1.03E-15
SUBURBAN	Tau	p value
Mean	0.63	8.61E-08
Median	0.61	2.53E-07
Std Dev	0.49	3.17E-05
Min	0.39	8.46E-04
Max	0.80	1.25E-11
RURAL	Tau	p value
Mean	0.98	1.24E-16
Median	0.97	1.99E-16
Std Dev	0.56	2.26E-06
Min	0.85	7.41E-13
Max	0.95	8.13E-16

However, this was not true for the Suburban class. These results drive interesting possibilities for follow on research. Why the Rural and Urban class align and why did

the Suburban class only partially align? If we identify the concept learned that drives a prediction value in one network, can we forecast the value in the other network?

If so, are their potential applications if this can be viewed as a semi-rigid ensemble of neural networks. That is, can we use this type of ensemble for different concurrent sub-tasks, then blending the results to produce integrated results in a superset, as if from a larger network. To paraphrase the news industry, an ensemble of neural networks, just might have legs.

Table 5 Two sets of Urban Logit values from two different datasets

Urban Logit Values of Remote Sensing Data from 35 VGG16 Models Trained only on NAIP Data

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
73.8	24.9	63.6	46.0	127.4	28.9	25.6	1.8	-1.9	32.1	25.0	44.8	46.1	60.6	49.0	76.2	49.7	40.4	62.4	60.9	22.1	16.2	41.6	8.5	10.3	33.8	115.7	170.0	9.4	38.3	15.1	76.9	39.0	75.4	12.3
73.8	24.8	63.5	45.3	127.3	28.5	25.3	2.0	-1.9	32.0	24.8	44.6	45.9	60.5	48.9	76.1	49.6	40.4	61.6	60.5	22.0	16.2	41.3	8.4	10.1	34.0	115.3	169.4	9.5	38.3	15.3	76.5	38.8	75.3	12.2
73.3	24.7	63.3	45.0	132.1	22.8	27.3	-1.3	-1.8	31.2	24.4	45.2	45.5	62.1	50.6	72.5	48.3	40.1	59.0	57.0	21.5	16.1	41.1	8.4	10.2	33.0	114.1	169.6	9.0	36.9	14.3	77.1	38.0	74.0	11.7
73.7	24.8	62.9	43.9	131.5	25.4	28.2	0.2	-1.8	31.3	24.9	44.0	44.5	61.9	50.0	74.6	48.0	39.2	59.7	58.8	21.1	15.9	40.7	8.2	9.1	32.6	113.6	169.0	9.3	36.3	14.9	75.0	37.0	74.5	11.9
71.9	24.5	61.7	46.1	128.2	21.1	25.8	0.3	-1.8	30.9	24.1	44.4	45.4	60.3	49.1	72.7	48.6	39.5	60.8	58.0	21.9	15.8	40.8	8.2	10.6	31.9	114.8	166.9	8.5	37.3	13.9	76.4	38.4	73.5	11.6
72.9	24.6	62.0	44.4	127.3	23.7	27.0	1.1	-1.8	31.3	24.6	43.8	44.4	60.6	47.7	73.7	48.0	39.6	59.8	58.3	21.3	15.9	40.9	8.3	9.6	32.6	113.4	168.2	9.1	36.4	15.0	74.6	36.8	71.9	11.8
72.5	24.2	63.0	43.4	131.6	23.6	26.3	1.4	-1.8	30.6	24.2	44.0	44.3	61.0	48.8	75.1	47.6	39.1	58.5	58.0	21.1	16.1	40.7	7.8	8.8	33.0	113.5	165.1	9.2	36.5	15.0	73.7	36.6	73.3	11.8
71.8	24.3	61.7	47.5	121.1	20.6	25.3	2.3	-1.8	31.3	24.3	43.6	44.9	58.1	47.8	74.1	47.3	39.5	62.6	60.4	21.9	15.5	41.3	8.3	10.9	31.8	115.9	161.1	9.3	38.6	15.2	74.3	39.0	73.1	11.7
72.8	24.5	61.8	43.8	133.1	22.4	26.1	-2.2	-1.8	31.1	24.4	43.8	43.9	62.0	50.6	72.3	47.8	38.6	57.6	57.2	20.8	15.7	40.9	8.1	8.7	32.3	113.0	169.1	9.1	34.9	14.6	75.3	36.0	73.2	11.5
72.2	23.9	61.8	43.9	130.3	25.0	25.3	0.2	-1.8	30.6	24.8	44.6	44.1	60.1	49.2	74.6	47.8	38.0	58.7	57.4	20.4	16.3	40.3	7.7	9.1	32.9	113.4	164.5	8.5	36.7	14.1	73.1	36.2	75.5	11.7

Urban Logit Values of Synthetic Data from 35 VGG16 Models, still only trained on NAIP Data, but running inference on Synthetic Data

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
83.2	28.7	69.6	51.0	156.9	38.9	33.9	2.5	-2.1	36.0	26.9	52.0	50.2	75.1	62.6	83.0	54.1	48.0	67.6	65.0	23.6	17.4	44.7	9.2	12.0	35.6	117.9	193.1	10.4	42.4	17.6	86.9	42.0	82.0	13.2
81.7	28.3	68.9	52.1	151.5	38.8	37.9	2.2	-2.1	36.3	26.4	51.3	50.1	74.1	59.4	83.2	53.9	48.2	69.4	65.8	24.0	17.4	44.5	9.2	12.4	36.0	119.4	188.7	10.4	43.4	17.1	85.9	43.2	80.2	13.4
83.1	28.5	68.8	52.2	152.1	38.4	32.7	2.9	-2.1	36.1	26.6	51.7	50.3	73.6	60.8	81.8	54.3	48.4	68.4	64.5	24.2	17.1	45.0	9.5	12.6	35.5	118.1	192.2	10.5	42.9	17.3	86.3	42.8	81.2	13.2
81.5	28.3	67.5	51.6	151.1	38.1	33.3	2.3	-2.1	36.0	26.4	51.1	50.1	73.1	60.2	82.3	53.9	47.5	68.3	64.3	23.9	17.1	44.2	9.4	12.3	35.9	118.4	189.5	10.3	42.3	16.9	86.0	42.7	79.4	13.1
81.3	28.3	68.2	50.6	151.5	38.7	32.1	2.4	-2.1	35.7	26.4	51.3	49.8	74.8	58.9	81.9	53.8	46.7	66.8	63.4	23.5	17.2	44.0	9.3	11.9	35.8	117.4	188.5	10.0	41.8	17.3	86.2	42.3	80.8	13.1
76.8	28.0	68.1	57.0	129.1	31.7	27.6	1.4	-2.1	36.4	26.0	50.8	52.0	64.2	56.7	83.3	53.6	47.9	73.9	67.9	24.8	16.7	46.2	9.1	14.9	34.2	122.9	183.0	10.2	44.9	16.3	87.7	46.5	79.2	13.4
78.8	27.3	67.2	52.6	141.9	34.0	28.5	2.1	-2.0	34.4	25.5	52.7	49.6	69.0	57.8	79.3	52.4	47.1	66.8	64.9	23.6	16.6	43.6	9.1	13.2	34.4	119.2	182.2	9.6	43.2	15.9	83.2	42.5	81.8	12.4
78.2	28.2	67.0	53.7	135.8	30.9	29.5	2.4	-2.0	35.3	25.8	52.2	49.9	67.8	55.2	81.1	52.1	46.4	68.4	65.2	23.9	16.2	45.0	9.4	13.7	34.2	120.6	183.3	9.8	43.2	16.7	85.7	44.1	78.9	12.8
76.7	27.6	66.9	54.8	138.5	29.3	28.3	1.5	-1.9	34.5	25.3	52.8	50.5	67.3	56.8	79.7	52.5	47.1	68.6	65.2	23.7	16.4	44.8	9.2	14.1	32.9	119.9	179.5	9.5	42.8	15.8	85.8	44.1	80.4	12.6
76.1	27.5	66.8	54.4	137.8	30.9	28.9	0.8	-2.0	34.2	25.6	52.1	50.3	66.7	57.0	82.4	52.3	46.4	68.4	65.7	24.0	16.4	44.7	9.0	13.9	33.6	120.7	176.7	9.1	43.4	15.6	85.8	44.0	80.4	12.7

APPENDIX 1 SUMMARY TCAV FOR 35 VGG16 NETWORKS ON NAIP

Summary TCAV results from VGG16 networks trained only on NAIP data

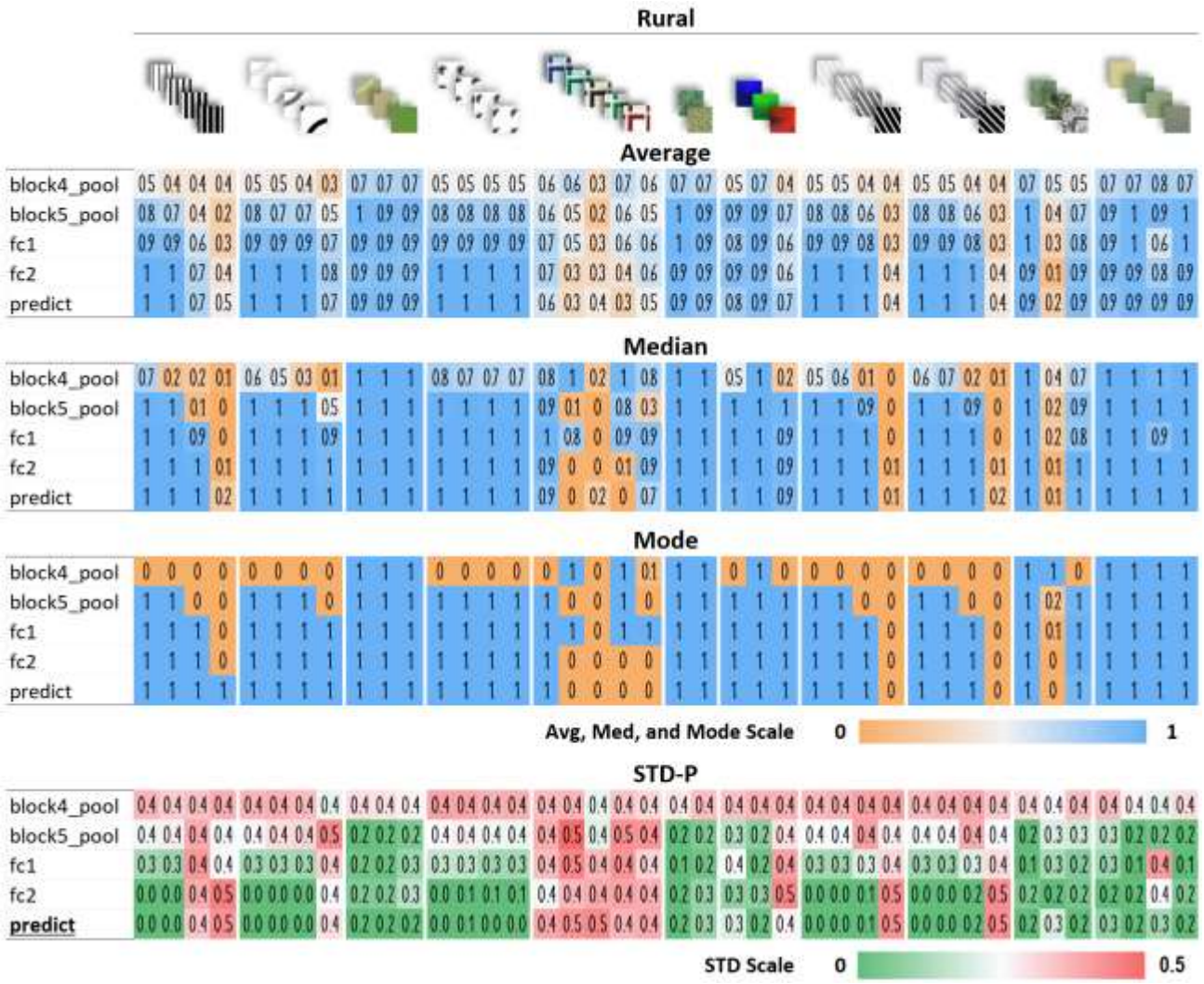


Figure 37 Rural Class Summary Statistics of First 40 Concepts across 35 NAIP VGG16 models

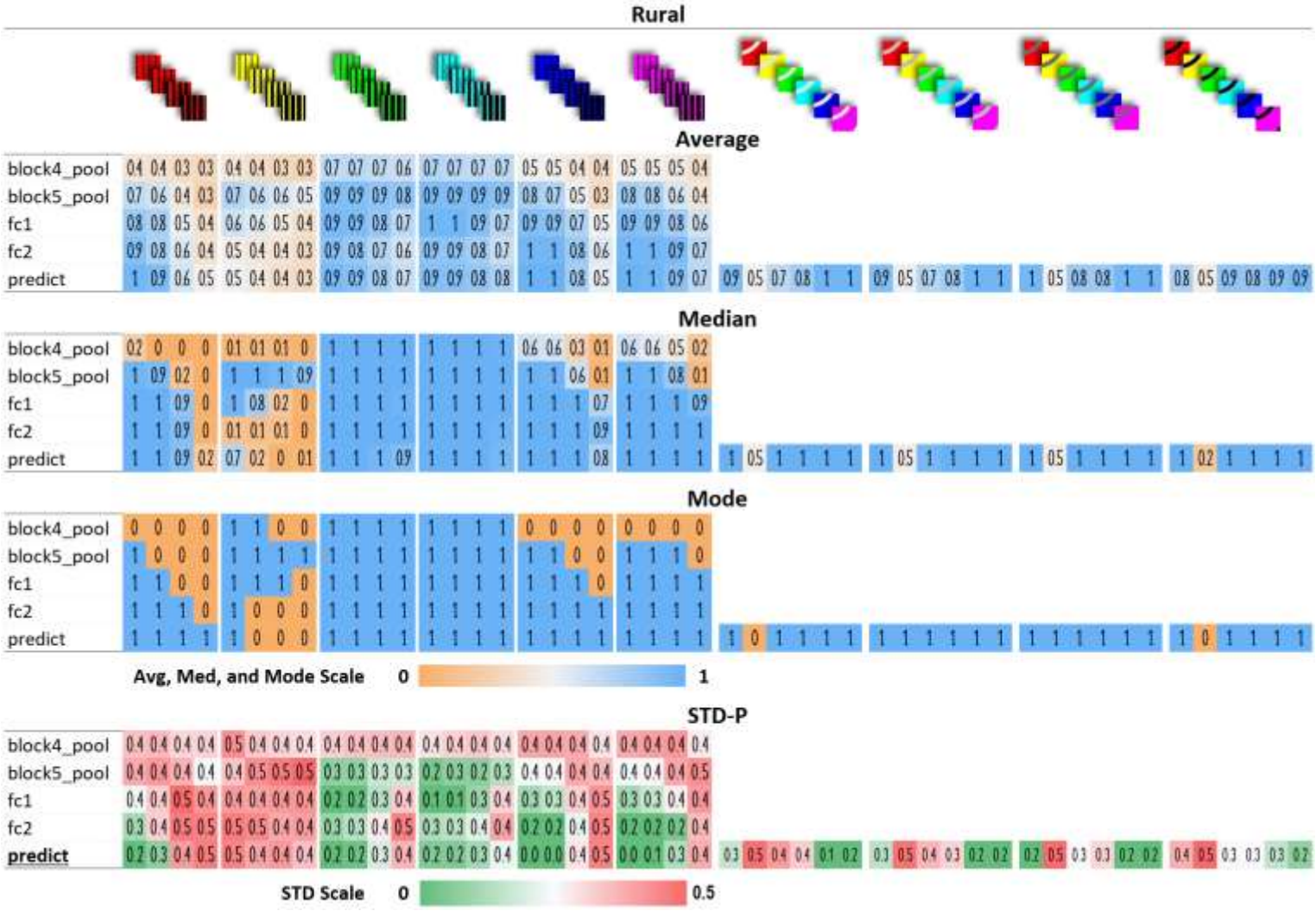


Figure 38 Rural Class Summary Statistics of Second 48 Concepts across 35 NAIP VGG16 models

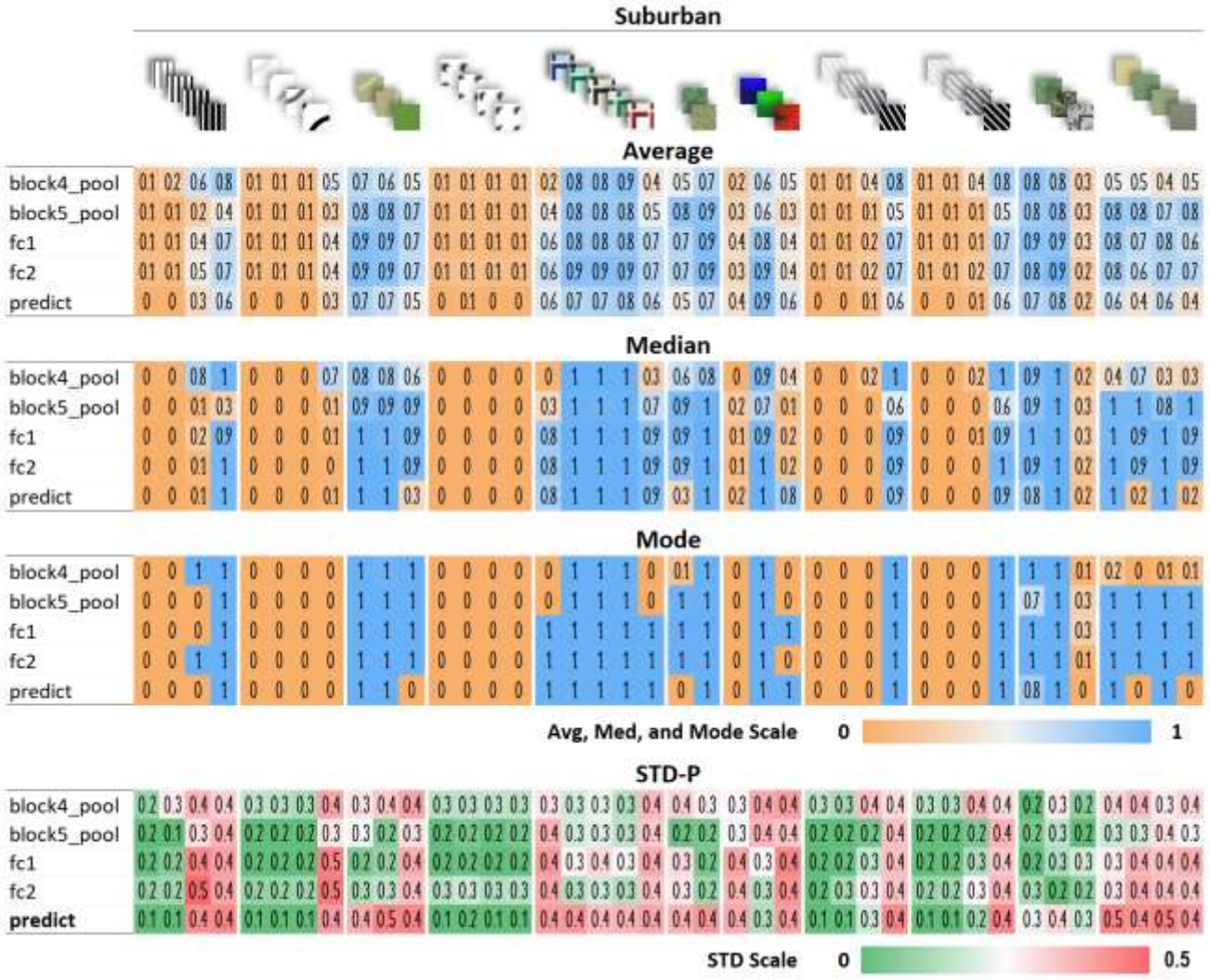


Figure 39 Suburban Class Summary Statistics of First 40 Concepts across 35 NAIP VGG16 models

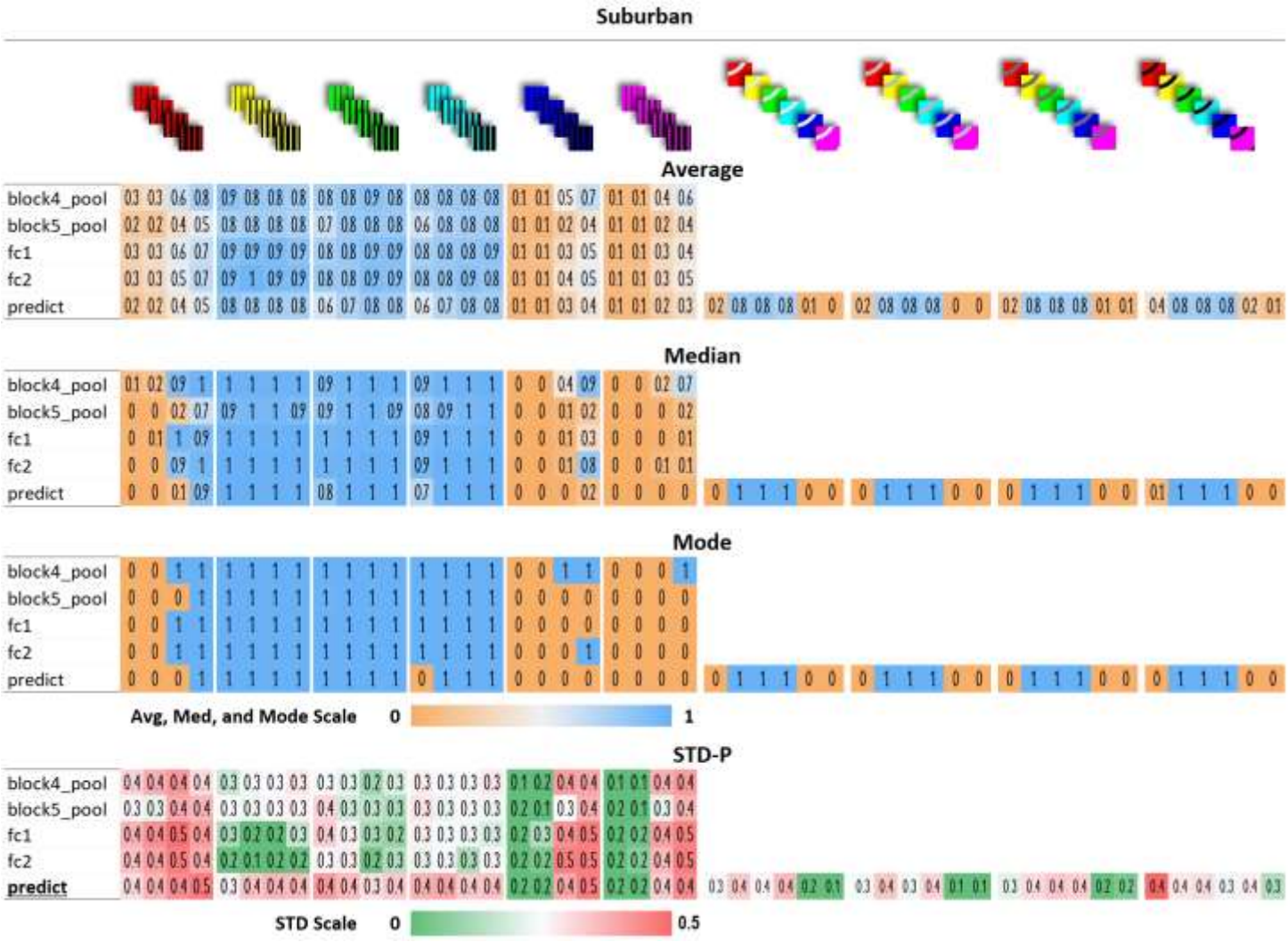


Figure 40 Suburban Class Summary Statistics of Second 48 Concepts across 35 NAIP VGG16 models

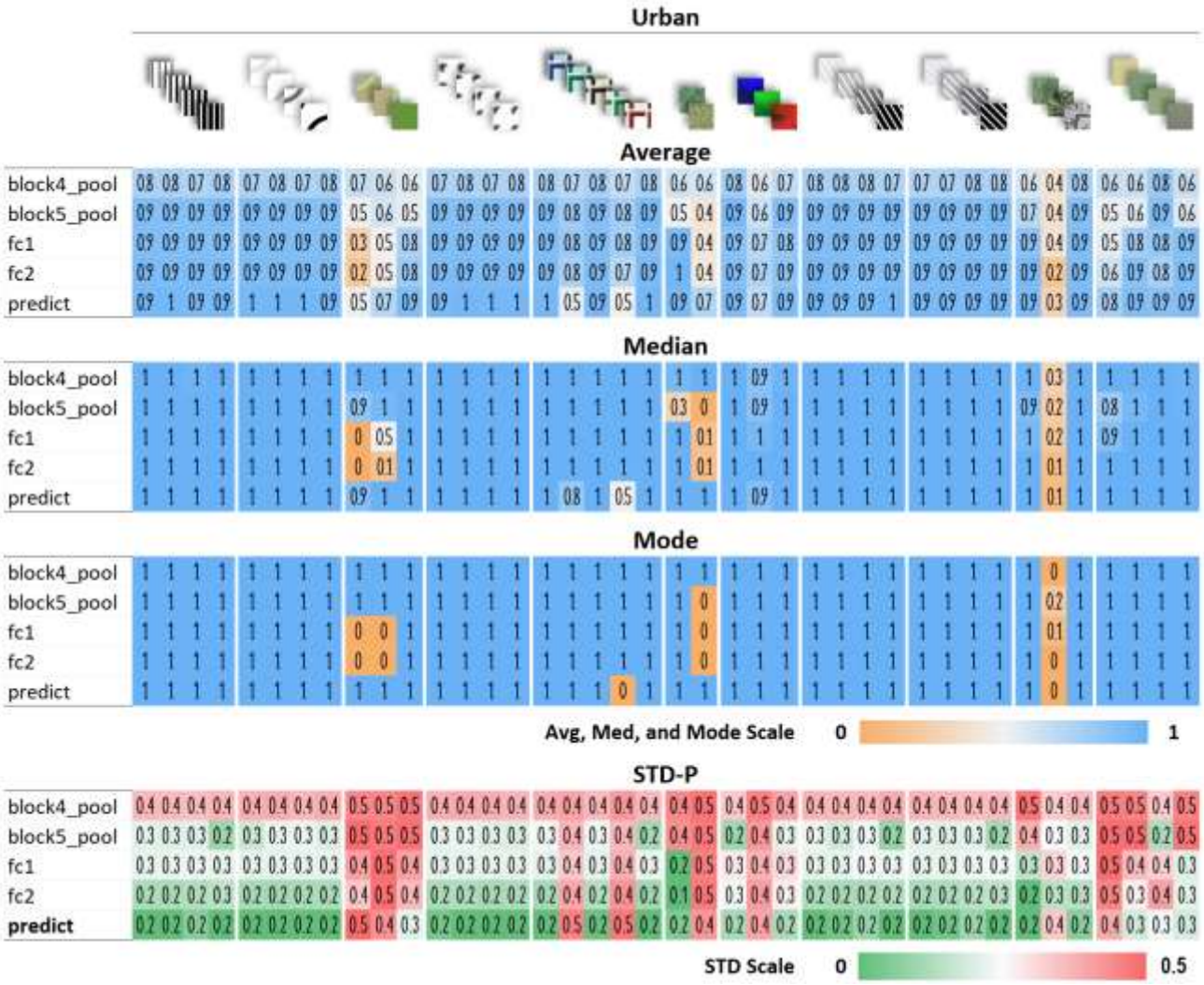


Figure 41 Urban Class Summary Statistics of First 40 Concepts across 35 NAIP VGG16 models

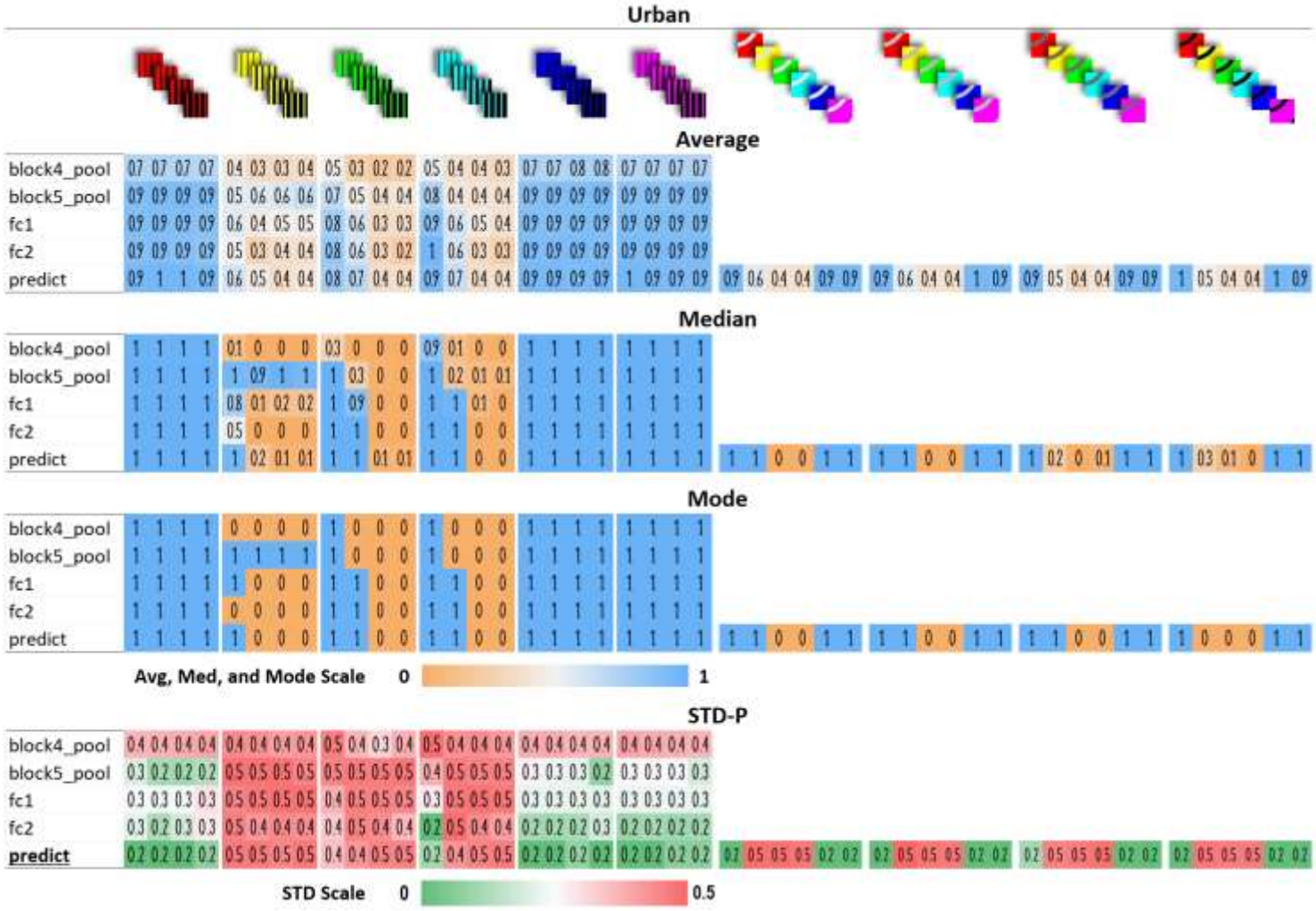


Figure 42 Urban Class Summary Statistics of Second 48 Concepts across 35 NAIP VGG16 models

APPENDIX 2 TCAV RESULTS FROM GRADUAL UNFREEZING

This Appendix contains the TCAV results from Gradual Unfreezing Experiments

Figure 43 Gradual Unfreezing Experiments using G-RSU v1 Synth Dataset, Rural Class

Figure 44 Gradual Unfreezing Experiments using G-RSU v2 Synth Dataset, Rural Class

Figure 45 Gradual Unfreezing Experiments using G-RSU v1 Synth Dataset, Suburban Class

Figure 46 Gradual Unfreezing Experiments using G-RSU v2 Synth Dataset, Suburban Class

Figure 47 Gradual Unfreezing Experiments using G-RSU v1 Synth Dataset, Urban Class

Figure 48 Gradual Unfreezing Experiments using G-RSU v2 Synth Dataset, Urban Class

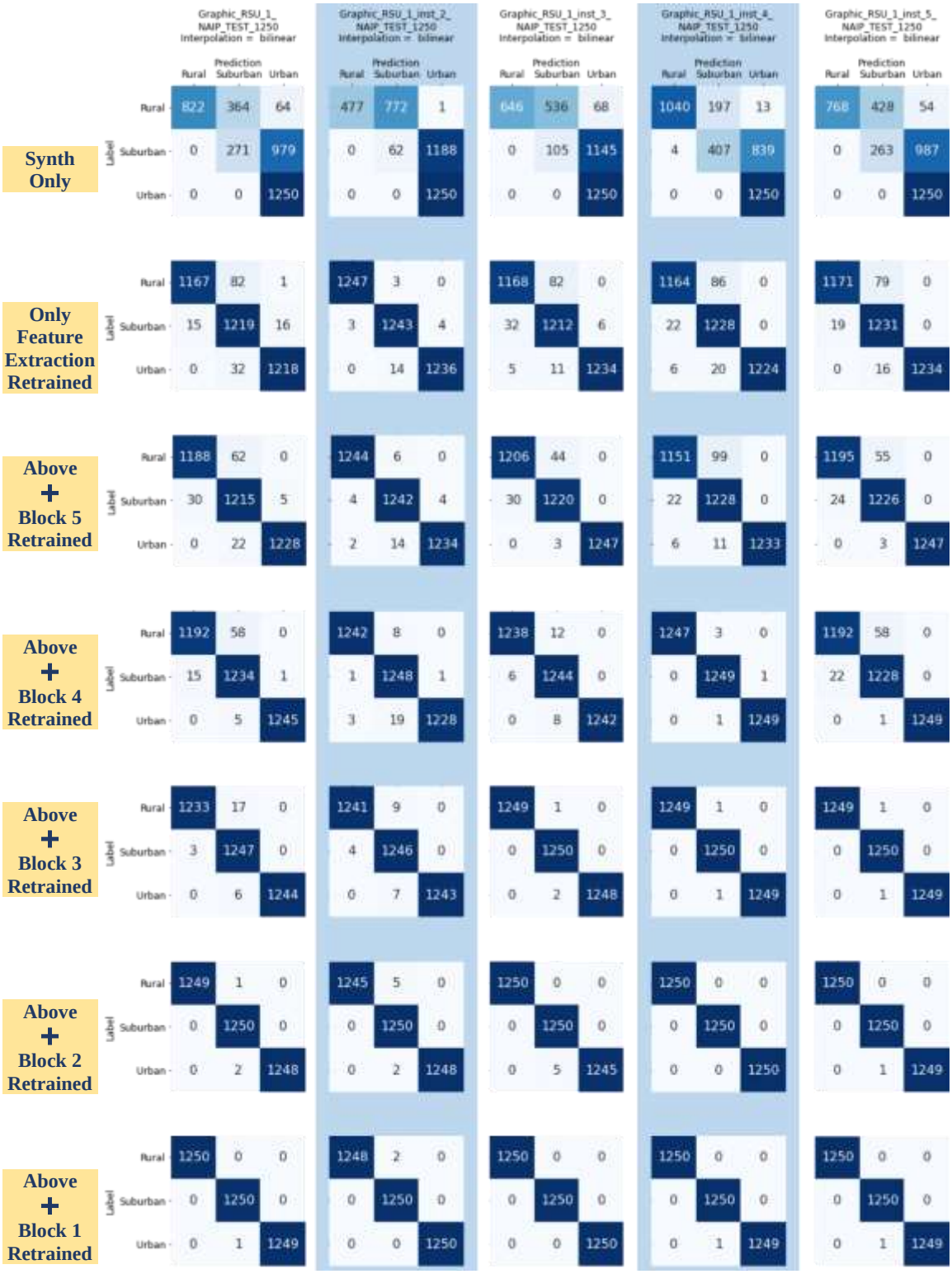


Figure 49 Confusion Matrices from Gradual Unfreezing -Synthetic G-RSU, Version 1

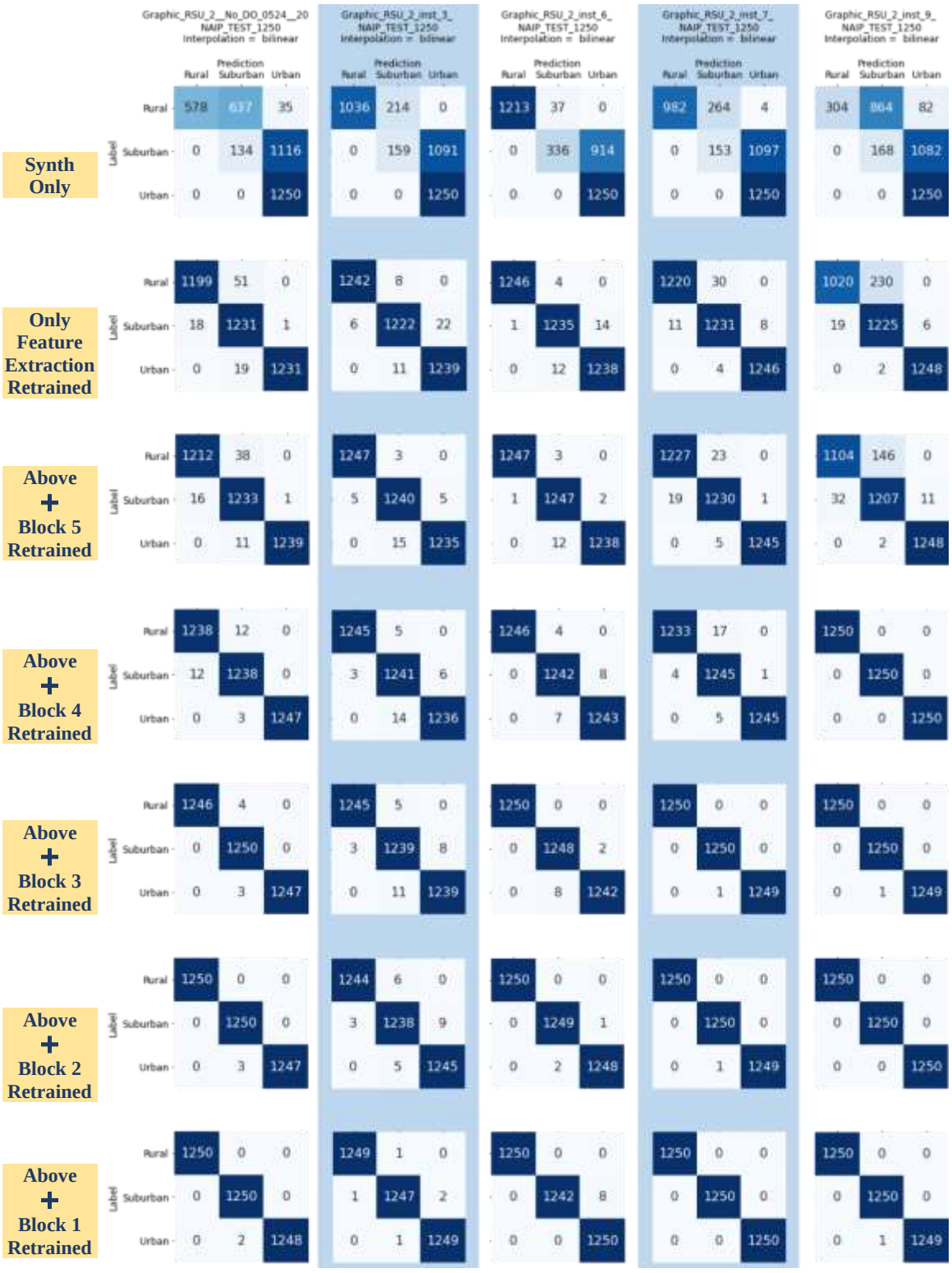


Figure 50 Confusion Matrices from Gradual Unfreezing -Synthetic G-RSU, Version 2

APPENDIX 3 TCAV 35 NAIP VGG16 RESULT HEATMAPS

This appendix contains TCAV results from 35 VGG16 Networks trained only on NAIP data, these are presented as Heatmaps.

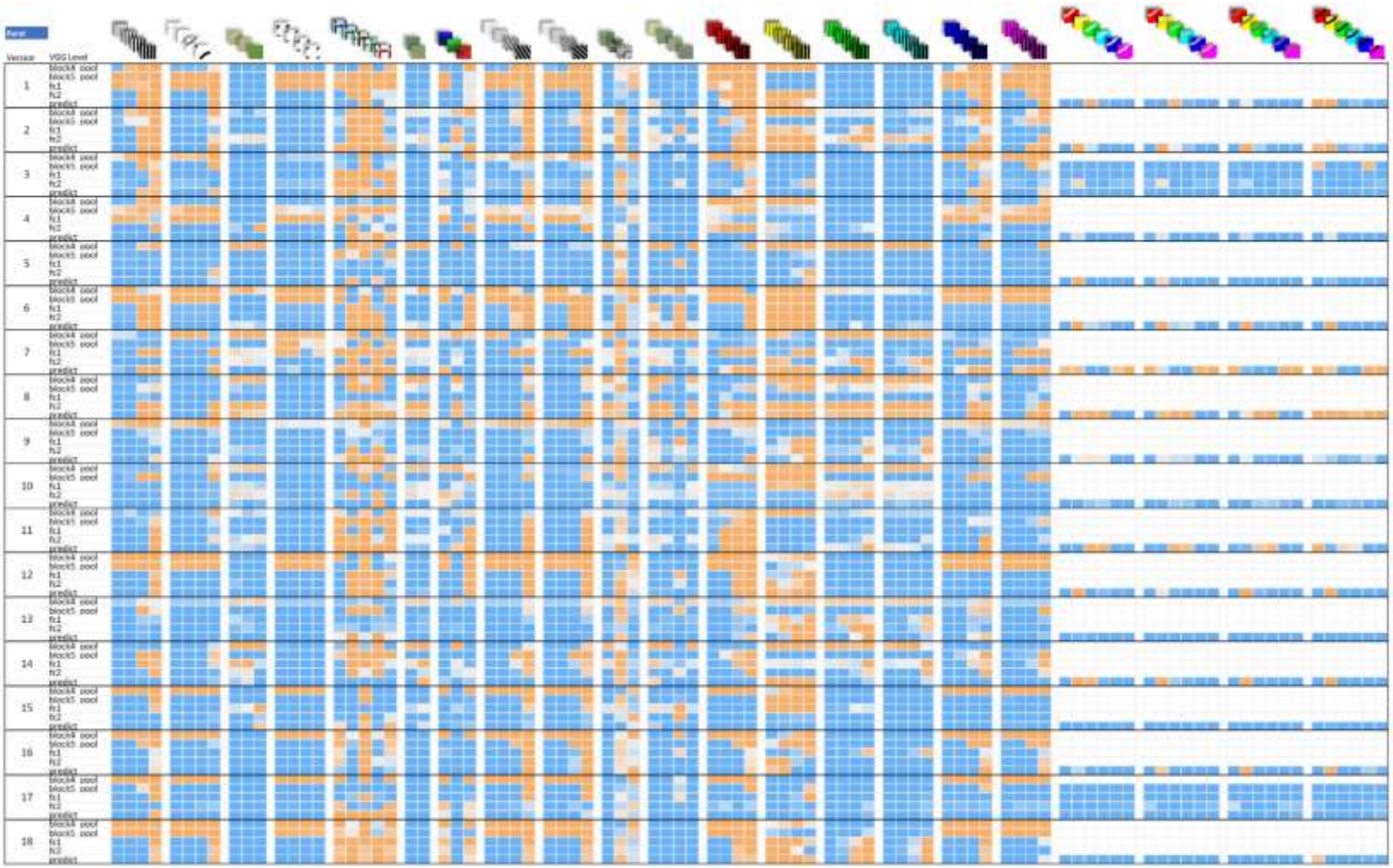


Figure 51 TCAV Results from Rural Class, instances 1 to 18

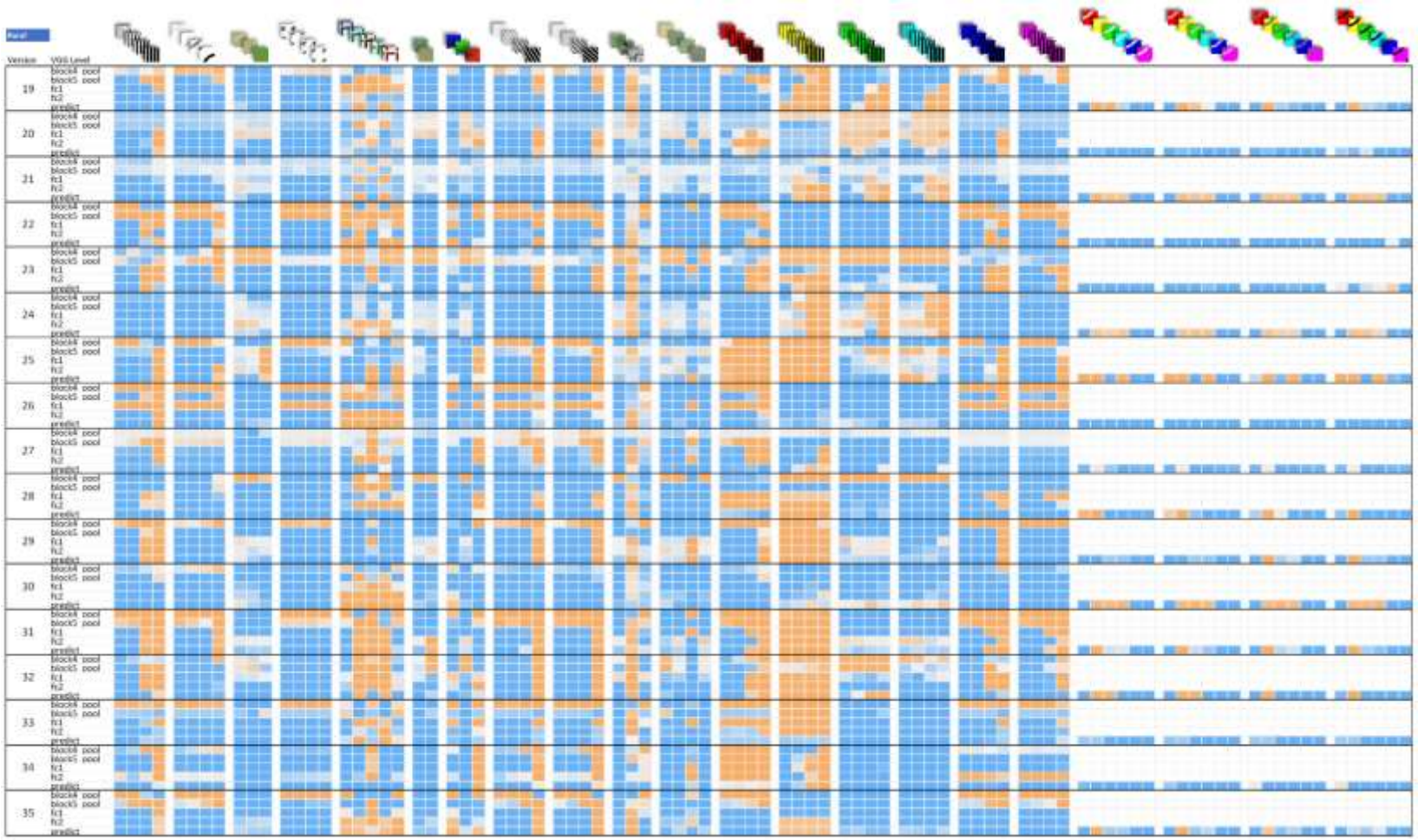


Figure 52 TCAV Results from Rural Class, instances 19 to 35

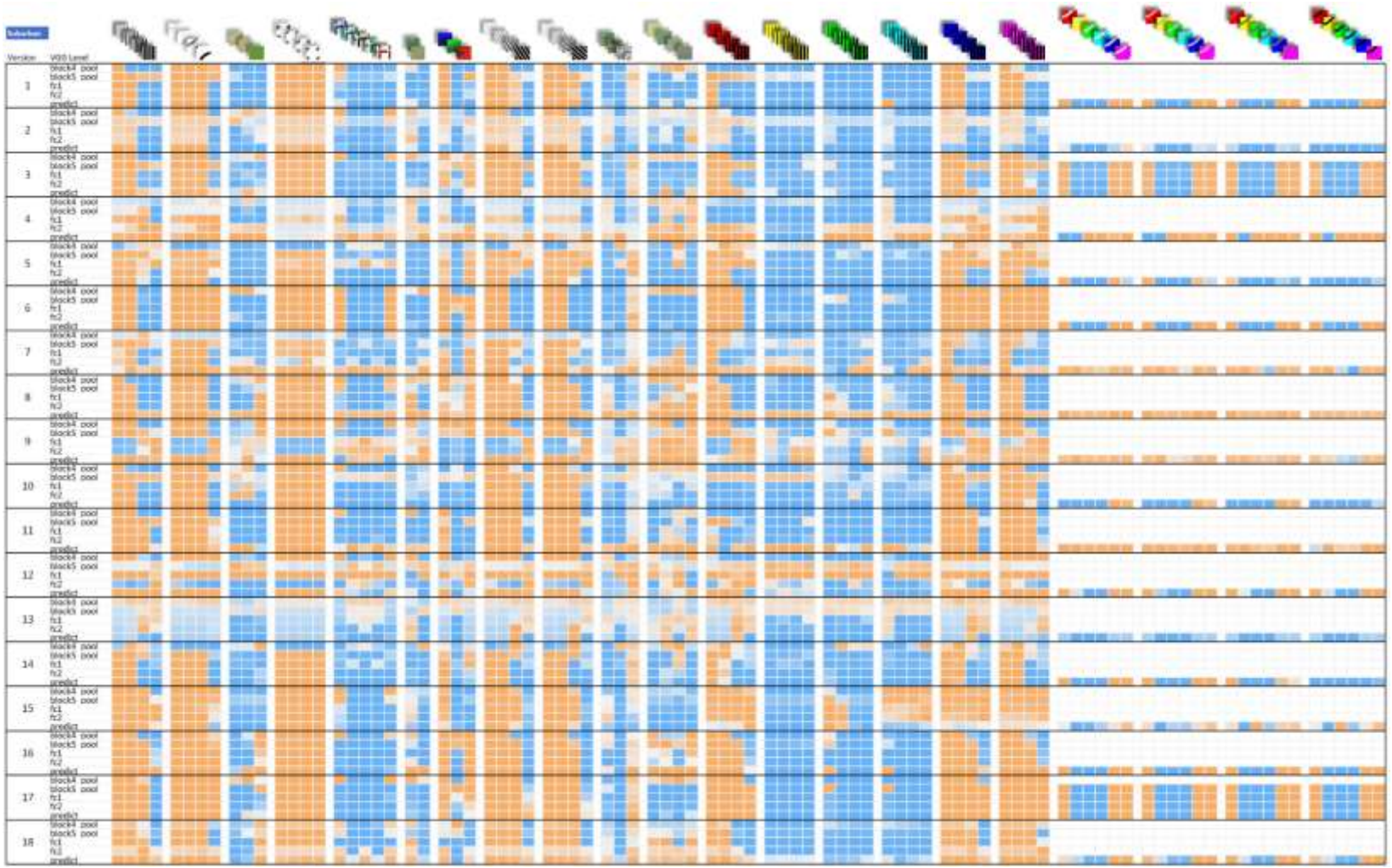


Figure 53 TCAV Results from Suburban Class, instances 1 to 18

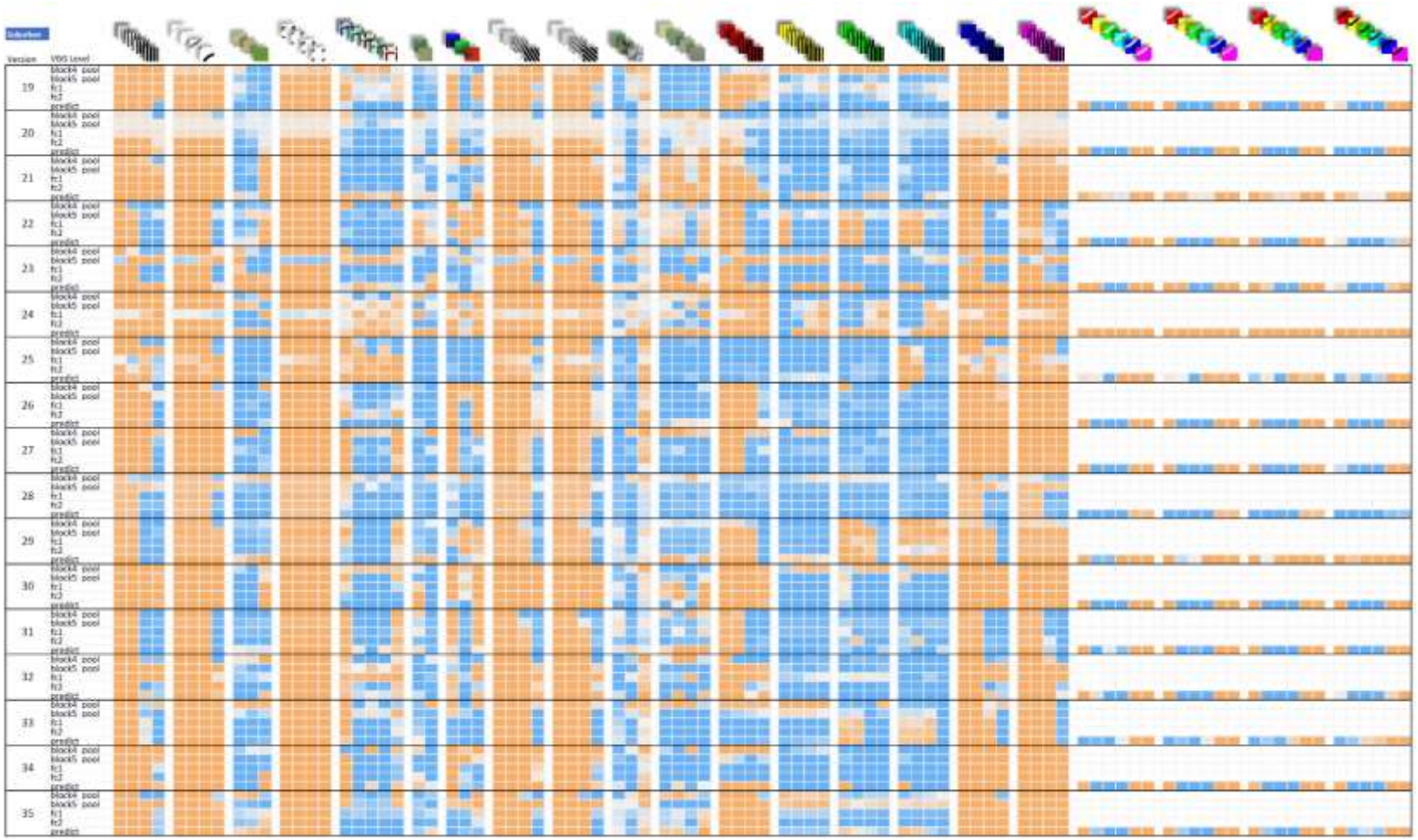


Figure 54 TCAV Results from Suburban Class, instances 19 to 35

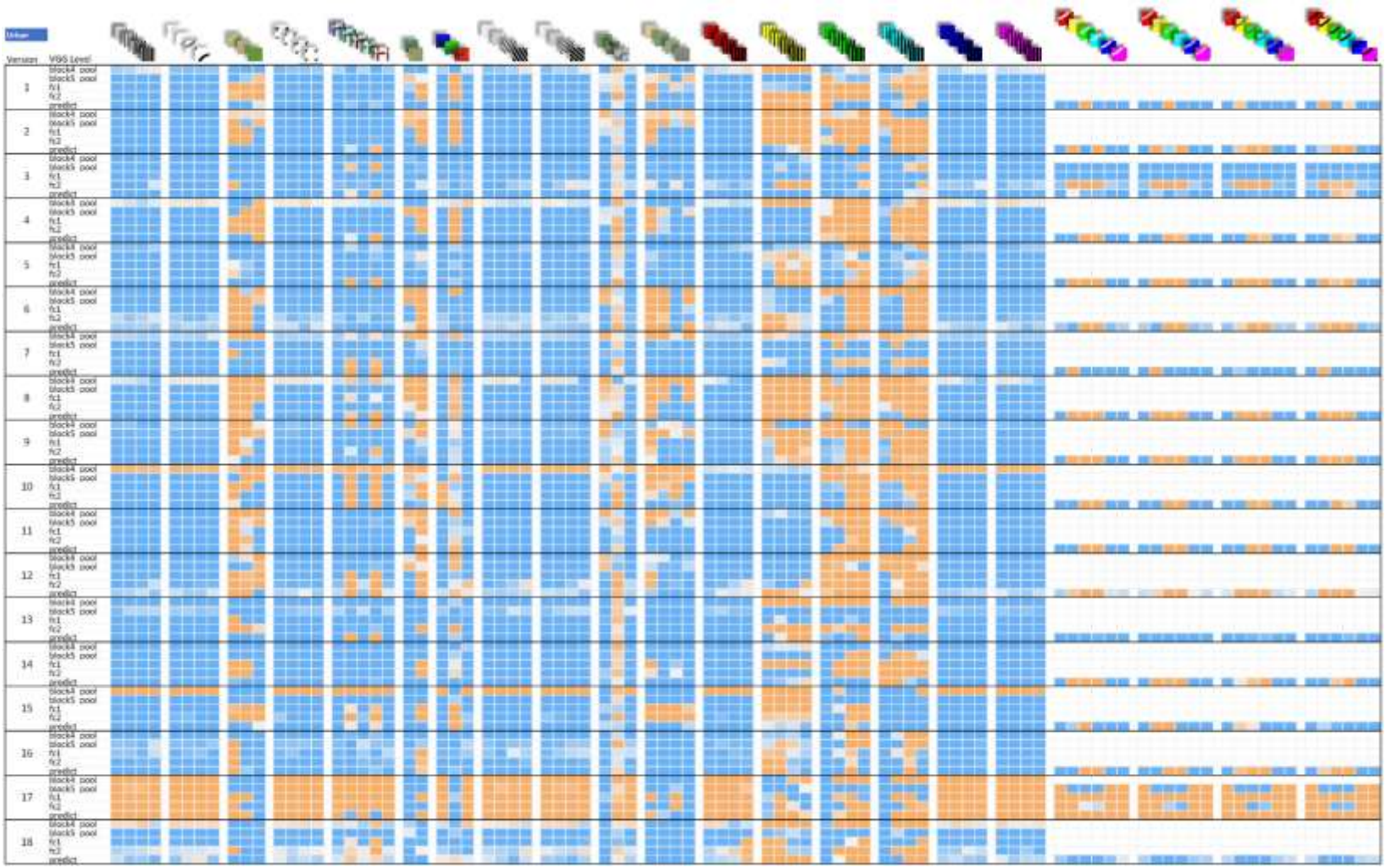


Figure 55 TCAV Results from Urban Class, instances 1 to 18

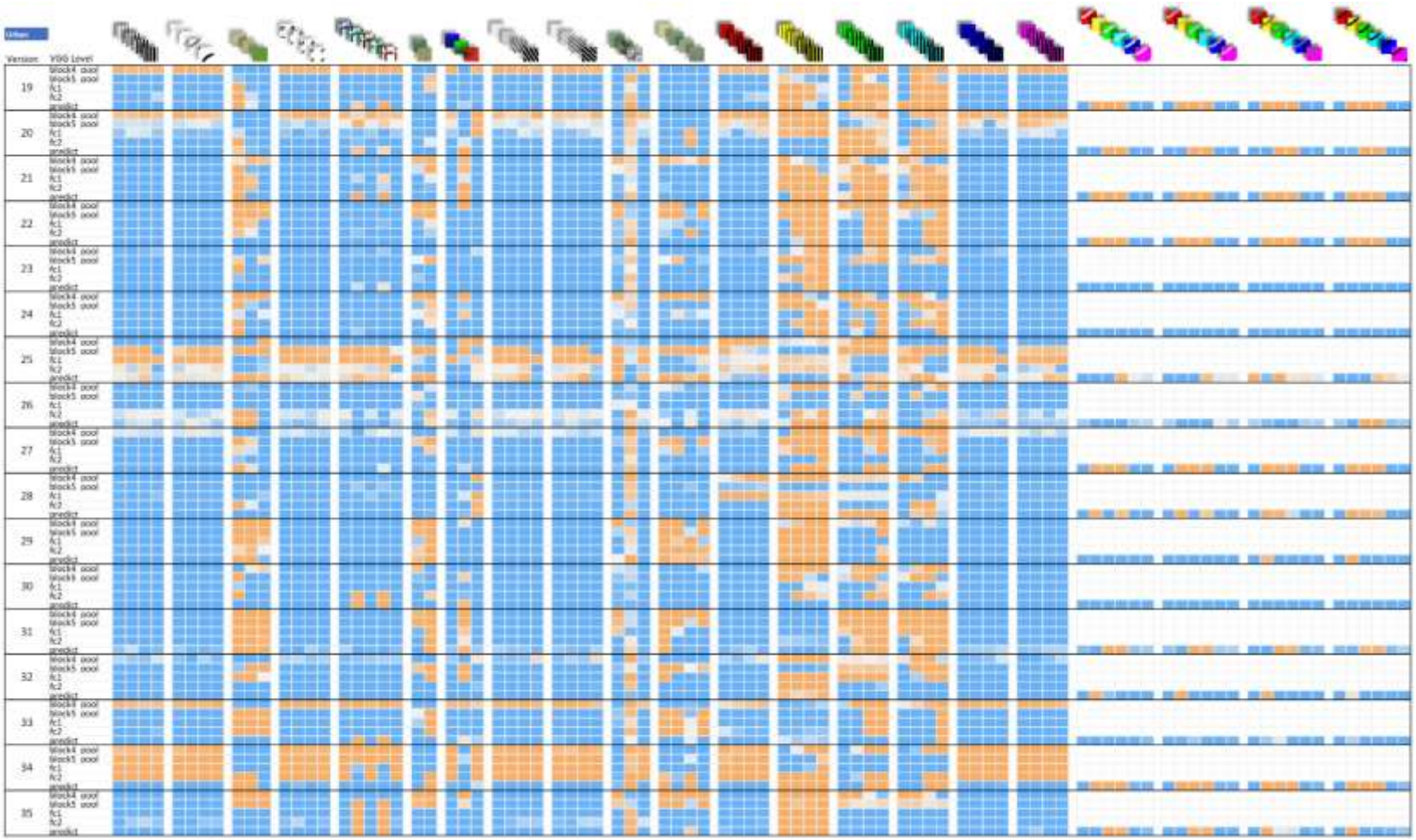


Figure 56 TCAV Results from Urban Class, instances 19 to 35

APPENDIX 4 TCAV IMAGE CONCEPTS USED AND RAW DATA

This appendix shows example TCAV concepts and an example TCAV results from one class of one VGG16 network trained on NAIP data.

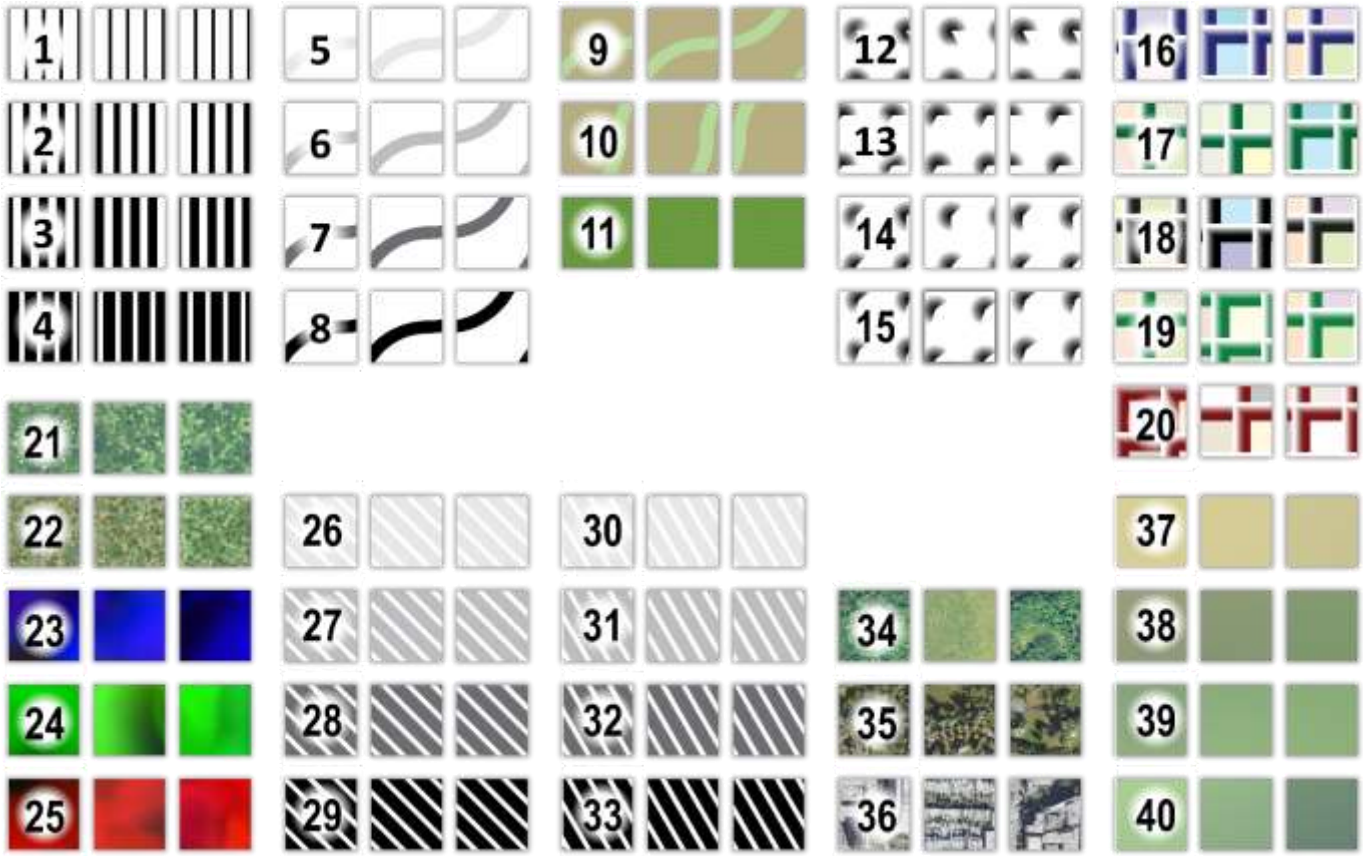


Figure 57 TCAV Concepts 1 to 40

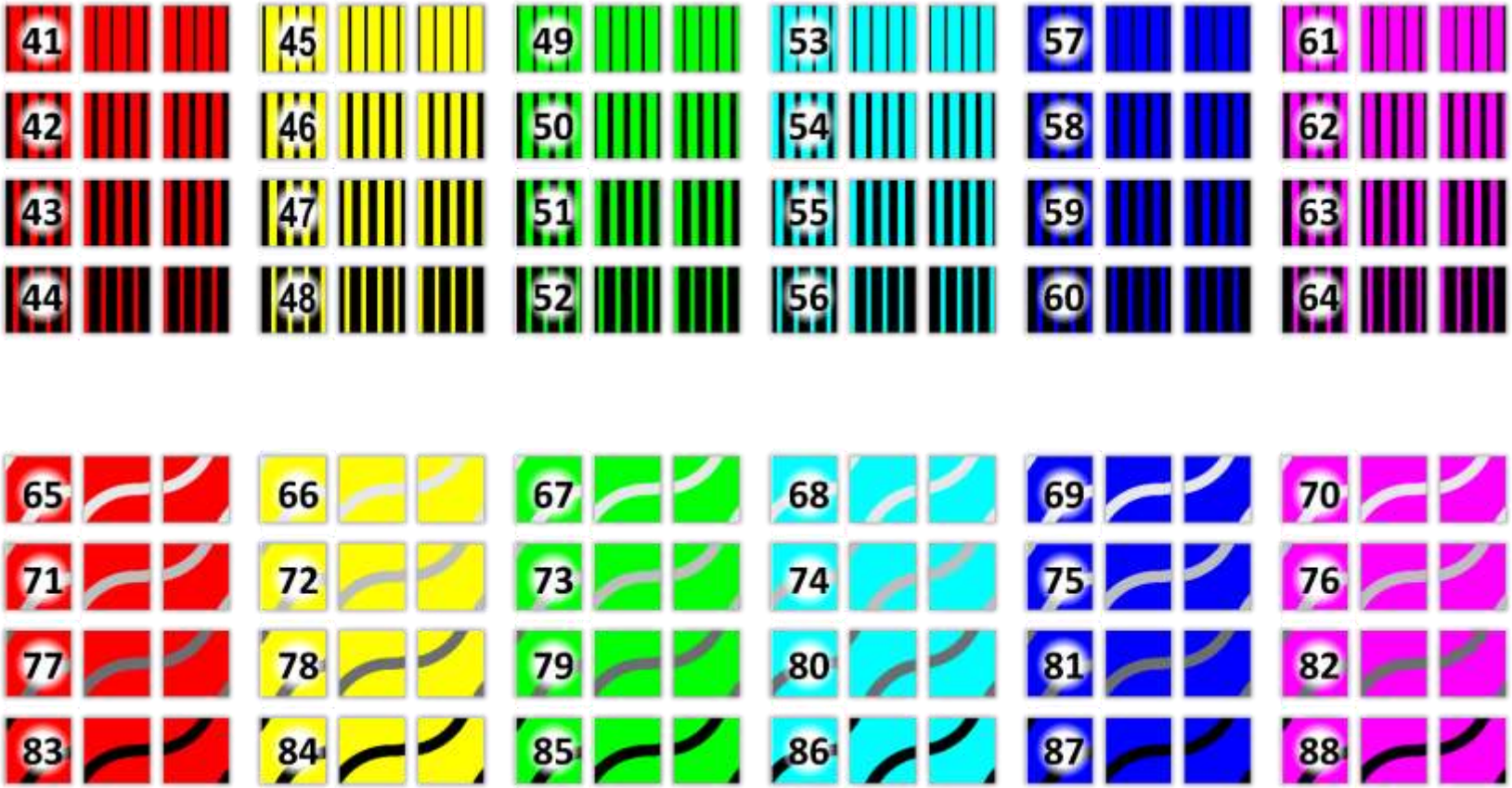


Figure 58 TCAV Concepts 41 to 88.

Table 6 TCAV Concept Key for Appendix Table

1 10W_WHT	23 Medium_Blue_Color_Blobs	45 Clr_Strp_Ylw__10W	67 Arcs_040_K010__Grn
2 20W_WHT	24 Medium_Green_Color_Blobs	46 Clr_Strp_Ylw__20W	68 Arcs_040_K010__BGn
3 30W_WHT	25 Medium_Red_Color_Blobs	47 Clr_Strp_Ylw__30W	69 Arcs_040_K010__Blu
4 40W_WHT	26 Stripes_40_30deg_K010	48 Clr_Strp_Ylw__40W	70 Arcs_040_K010__Vlt
5 Arcs_040_K010	27 Stripes_40_30deg_K030	49 Clr_Strp_Grn__10W	71 Arcs_040_K030__Red
6 Arcs_040_K030	28 Stripes_40_30deg_K070	50 Clr_Strp_Grn__20W	72 Arcs_040_K030__Ylw
7 Arcs_040_K070	29 Stripes_40_30deg_K100	51 Clr_Strp_Grn__30W	73 Arcs_040_K030__Grn
8 Arcs_040_K100	30 Stripes_40_45deg_K010	52 Clr_Strp_Grn__40W	74 Arcs_040_K030__BGn
9 Arcs_WI_Colors_1	31 Stripes_40_45deg_K030	53 Clr_Strp_BGn__10W	75 Arcs_040_K030__Blu
10 Arcs_WI_Colors_2	32 Stripes_40_45deg_K070	54 Clr_Strp_BGn__20W	76 Arcs_040_K030__Vlt
11 Fishscales_1	33 Stripes_40_45deg_K100	55 Clr_Strp_BGn__30W	77 Arcs_040_K070__Red
12 FR_ANG_RADL_GRDNT_01_TRI	34 Rural_as_Concept_5K	56 Clr_Strp_BGn__40W	78 Arcs_040_K070__Ylw
13 FR_ANG_RADL_GRDNT_02_HEX	35 Suburban_as_Concept_5K	57 Clr_Strp_Blu__10W	79 Arcs_040_K070__Grn
14 FR_ANG_RADL_GRDNT_03_OCT	36 Urban_as_Concept_5K	58 Clr_Strp_Blu__20W	80 Arcs_040_K070__BGn
15 FR_ANG_RADL_GRDNT_04_DEC	37 WI_Col_Wsh_001-200	59 Clr_Strp_Blu__30W	81 Arcs_040_K070__Blu
16 FR_BLU_W100_OP100_G-K00	38 WI_Col_Wsh_1001-1200	60 Clr_Strp_Blu__40W	82 Arcs_040_K070__Vlt
17 FR_DK_GRN_W100_OP100_G-K00	39 WI_Col_Wsh_1501-1700	61 Clr_Strp_Vlt__10W	83 Arcs_040_K100__Red
18 FR_Gray_W100_OP90-K00	40 WI_Col_Wsh_501-700	62 Clr_Strp_Vlt__20W	84 Arcs_040_K100__Ylw
19 FR_GRN_W100_OP100_G-K00	41 Clr_Strp_Red__10W	63 Clr_Strp_Vlt__30W	85 Arcs_040_K100_Grn
20 FR_RED_W100_OP100_G-K00	42 Clr_Strp_Red__20W	64 Clr_Strp_Vlt__40W	86 Arcs_040_K100__BGn
21 Graphical_Rural_WI	43 Clr_Strp_Red__30W	65 Arcs_040_K010__Red	87 Arcs_040_K100__Blu
22 Graphical_Rural_WI_v2	44 Clr_Strp_Red__40W	66 Arcs_040_K010__Ylw	88 Arcs_040_K100__Vlt

Table 7 Example TCAV Values from just one class of a single VGG16 instance

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20220321__1115	Rural	_1118	5	1	10W_WHT	block4_pool	0.97	0.06	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	1	10W_WHT	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	1	10W_WHT	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	1	10W_WHT	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	1	10W_WHT	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	2	20W_WHT	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	2	20W_WHT	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	2	20W_WHT	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	2	20W_WHT	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	2	20W_WHT	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	3	30W_WHT	block4_pool	0.25	0.37	0.52	0.45	0.006	sig.
20211209__1556	Rural	_1118	5	3	30W_WHT	block5_pool	0.98	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	3	30W_WHT	fc1	0.99	0.01	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	3	30W_WHT	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	3	30W_WHT	predict	1	0.01	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	4	40W_WHT	block4_pool	0.06	0.2	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	4	40W_WHT	block5_pool	0.98	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	4	40W_WHT	fc1	0.95	0.22	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	4	40W_WHT	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	4	40W_WHT	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	5	Arcs_040_K010	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	5	Arcs_040_K010	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	5	Arcs_040_K010	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	5	Arcs_040_K010	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	5	Arcs_040_K010	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	6	Arcs_040_K030	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	6	Arcs_040_K030	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	6	Arcs_040_K030	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	6	Arcs_040_K030	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	6	Arcs_040_K030	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	7	Arcs_040_K070	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	7	Arcs_040_K070	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	7	Arcs_040_K070	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	7	Arcs_040_K070	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	7	Arcs_040_K070	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	8	Arcs_040_K100	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	8	Arcs_040_K100	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	8	Arcs_040_K100	fc1	0.94	0.22	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	8	Arcs_040_K100	fc2	0.15	0.36	0.49	0.5	0.001	sig.
20211209__1556	Rural	_1118	5	8	Arcs_040_K100	predict	0.69	0.45	0.46	0.49	0.047	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20220321__1115	Rural	_1118	5	9	Arcs_WI_Colors_1	block4_pool	0.02	0.01	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	9	Arcs_WI_Colors_1	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	9	Arcs_WI_Colors_1	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	9	Arcs_WI_Colors_1	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	9	Arcs_WI_Colors_1	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	10	Arcs_WI_Colors_2	block4_pool	0.07	0.19	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	10	Arcs_WI_Colors_2	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	10	Arcs_WI_Colors_2	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	10	Arcs_WI_Colors_2	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	10	Arcs_WI_Colors_2	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	11	Fishscales_1	block4_pool	0.03	0.01	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	11	Fishscales_1	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	11	Fishscales_1	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	11	Fishscales_1	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	11	Fishscales_1	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	12	FR_ANG_RADL_GRDNT_01_TRI	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	12	FR_ANG_RADL_GRDNT_01_TRI	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	12	FR_ANG_RADL_GRDNT_01_TRI	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	12	FR_ANG_RADL_GRDNT_01_TRI	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	12	FR_ANG_RADL_GRDNT_01_TRI	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	13	FR_ANG_RADL_GRDNT_02_HEX	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	13	FR_ANG_RADL_GRDNT_02_HEX	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	13	FR_ANG_RADL_GRDNT_02_HEX	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	13	FR_ANG_RADL_GRDNT_02_HEX	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	13	FR_ANG_RADL_GRDNT_02_HEX	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	14	FR_ANG_RADL_GRDNT_03_OCT	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	14	FR_ANG_RADL_GRDNT_03_OCT	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	14	FR_ANG_RADL_GRDNT_03_OCT	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	14	FR_ANG_RADL_GRDNT_03_OCT	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	14	FR_ANG_RADL_GRDNT_03_OCT	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	15	FR_ANG_RADL_GRDNT_04_DEC	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	15	FR_ANG_RADL_GRDNT_04_DEC	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	15	FR_ANG_RADL_GRDNT_04_DEC	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	15	FR_ANG_RADL_GRDNT_04_DEC	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	15	FR_ANG_RADL_GRDNT_04_DEC	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	16	FR_BLU_W100_OP100_G-K00	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	16	FR_BLU_W100_OP100_G-K00	block5_pool	0	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	16	FR_BLU_W100_OP100_G-K00	fc1	0.85	0.35	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	16	FR_BLU_W100_OP100_G-K00	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	16	FR_BLU_W100_OP100_G-K00	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	17	FR_DK_GRN_W100_OP100_G-K00	block4_pool	0.03	0.01	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	17	FR_DK_GRN_W100_OP100_G-K00	block5_pool	0.42	0.47	0.5	0.49	0.459	not sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20211209__1556	Rural	_1118	5	17	FR_DK_GRN_W100_OP100_G-K00	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	17	FR_DK_GRN_W100_OP100_G-K00	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	17	FR_DK_GRN_W100_OP100_G-K00	predict	0.07	0.21	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	18	FR_Gray_W100_OP90-K00	block4_pool	0.1	0.19	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	18	FR_Gray_W100_OP90-K00	block5_pool	0	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	18	FR_Gray_W100_OP90-K00	fc1	0.9	0.3	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	18	FR_Gray_W100_OP90-K00	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	18	FR_Gray_W100_OP90-K00	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	19	FR_GRN_W100_OP100_G-K00	block4_pool	0.03	0.01	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	19	FR_GRN_W100_OP100_G-K00	block5_pool	0.9	0.29	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	19	FR_GRN_W100_OP100_G-K00	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	19	FR_GRN_W100_OP100_G-K00	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	19	FR_GRN_W100_OP100_G-K00	predict	0.21	0.39	0.46	0.49	0.014	sig.
20220321__1115	Rural	_1118	5	20	FR_RED_W100_OP100_G-K00	block4_pool	0.98	0.02	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	20	FR_RED_W100_OP100_G-K00	block5_pool	0	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	20	FR_RED_W100_OP100_G-K00	fc1	0.66	0.47	0.48	0.5	0.121	not sig.
20211209__1556	Rural	_1118	5	20	FR_RED_W100_OP100_G-K00	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	20	FR_RED_W100_OP100_G-K00	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	21	Graphical_Rural_WI	block4_pool	0.03	0.01	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	21	Graphical_Rural_WI	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	21	Graphical_Rural_WI	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	21	Graphical_Rural_WI	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	21	Graphical_Rural_WI	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	22	Graphical_Rural_WI_v2	block4_pool	0.03	0.03	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	22	Graphical_Rural_WI_v2	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	22	Graphical_Rural_WI_v2	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	22	Graphical_Rural_WI_v2	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	22	Graphical_Rural_WI_v2	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	23	Medium_Blue_Color_Blobs	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	23	Medium_Blue_Color_Blobs	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	23	Medium_Blue_Color_Blobs	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	23	Medium_Blue_Color_Blobs	fc2	0.99	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	23	Medium_Blue_Color_Blobs	predict	0.89	0.29	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	24	Medium_Green_Color_Blobs	block4_pool	0.02	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	24	Medium_Green_Color_Blobs	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	24	Medium_Green_Color_Blobs	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	24	Medium_Green_Color_Blobs	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	24	Medium_Green_Color_Blobs	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	25	Medium_Red_Color_Blobs	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	25	Medium_Red_Color_Blobs	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	25	Medium_Red_Color_Blobs	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	25	Medium_Red_Color_Blobs	fc2	1	0	0.49	0.5	0	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20211209__1556	Rural	_1118	5	25	Medium_Red_Color_Blobs	predict	0.99	0.01	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	26	Stripes_40_30deg_K010	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	26	Stripes_40_30deg_K010	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	26	Stripes_40_30deg_K010	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	26	Stripes_40_30deg_K010	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	26	Stripes_40_30deg_K010	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	27	Stripes_40_30deg_K030	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	27	Stripes_40_30deg_K030	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	27	Stripes_40_30deg_K030	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	27	Stripes_40_30deg_K030	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	27	Stripes_40_30deg_K030	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	28	Stripes_40_30deg_K070	block4_pool	0.58	0.42	0.52	0.45	0.504	not sig.
20211209__1556	Rural	_1118	5	28	Stripes_40_30deg_K070	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	28	Stripes_40_30deg_K070	fc1	1	0.01	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	28	Stripes_40_30deg_K070	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	28	Stripes_40_30deg_K070	predict	1	0.01	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	29	Stripes_40_30deg_K100	block4_pool	0.44	0.44	0.52	0.45	0.439	not sig.
20211209__1556	Rural	_1118	5	29	Stripes_40_30deg_K100	block5_pool	0.98	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	29	Stripes_40_30deg_K100	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	29	Stripes_40_30deg_K100	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	29	Stripes_40_30deg_K100	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	30	Stripes_40_45deg_K010	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	30	Stripes_40_45deg_K010	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	30	Stripes_40_45deg_K010	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	30	Stripes_40_45deg_K010	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	30	Stripes_40_45deg_K010	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	31	Stripes_40_45deg_K030	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	31	Stripes_40_45deg_K030	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	31	Stripes_40_45deg_K030	fc1	0.98	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	31	Stripes_40_45deg_K030	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	31	Stripes_40_45deg_K030	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	32	Stripes_40_45deg_K070	block4_pool	0.66	0.38	0.52	0.45	0.135	not sig.
20211209__1556	Rural	_1118	5	32	Stripes_40_45deg_K070	block5_pool	0.99	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	32	Stripes_40_45deg_K070	fc1	1	0.01	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	32	Stripes_40_45deg_K070	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	32	Stripes_40_45deg_K070	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	33	Stripes_40_45deg_K100	block4_pool	0.7	0.38	0.52	0.45	0.052	not sig.
20211209__1556	Rural	_1118	5	33	Stripes_40_45deg_K100	block5_pool	0.98	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	33	Stripes_40_45deg_K100	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	33	Stripes_40_45deg_K100	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	33	Stripes_40_45deg_K100	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	34	Rural_as_Concept_5K	block4_pool	0.02	0	0.52	0.45	0	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20211209__1556	Rural	_1118	5	34	Rural_as_Concept_5K	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	34	Rural_as_Concept_5K	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	34	Rural_as_Concept_5K	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	34	Rural_as_Concept_5K	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	35	Suburban_as_Concept_5K	block4_pool	0.49	0.46	0.52	0.45	0.828	not sig.
20211209__1556	Rural	_1118	5	35	Suburban_as_Concept_5K	block5_pool	0.01	0.01	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	35	Suburban_as_Concept_5K	fc1	0.46	0.49	0.48	0.5	0.857	not sig.
20211209__1556	Rural	_1118	5	35	Suburban_as_Concept_5K	fc2	0.2	0.4	0.49	0.5	0.006	sig.
20211209__1556	Rural	_1118	5	35	Suburban_as_Concept_5K	predict	0	0.01	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	36	Urban_as_Concept_5K	block4_pool	0.98	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	36	Urban_as_Concept_5K	block5_pool	0.99	0.01	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	36	Urban_as_Concept_5K	fc1	0.99	0.01	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	36	Urban_as_Concept_5K	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	36	Urban_as_Concept_5K	predict	0.99	0.01	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	37	WI_Col_Wsh_001-200	block4_pool	0.03	0.03	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	37	WI_Col_Wsh_001-200	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	37	WI_Col_Wsh_001-200	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	37	WI_Col_Wsh_001-200	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	37	WI_Col_Wsh_001-200	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	38	WI_Col_Wsh_1001-1200	block4_pool	0.03	0.01	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	38	WI_Col_Wsh_1001-1200	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	38	WI_Col_Wsh_1001-1200	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	38	WI_Col_Wsh_1001-1200	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	38	WI_Col_Wsh_1001-1200	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	39	WI_Col_Wsh_1501-1700	block4_pool	0.99	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	39	WI_Col_Wsh_1501-1700	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	39	WI_Col_Wsh_1501-1700	fc1	0.1	0.3	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	39	WI_Col_Wsh_1501-1700	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	39	WI_Col_Wsh_1501-1700	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	40	WI_Col_Wsh_501-700	block4_pool	0.02	0	0.52	0.45	0	sig.
20211209__1556	Rural	_1118	5	40	WI_Col_Wsh_501-700	block5_pool	1	0	0.5	0.49	0	sig.
20211209__1556	Rural	_1118	5	40	WI_Col_Wsh_501-700	fc1	1	0	0.48	0.5	0	sig.
20211209__1556	Rural	_1118	5	40	WI_Col_Wsh_501-700	fc2	1	0	0.49	0.5	0	sig.
20211209__1556	Rural	_1118	5	40	WI_Col_Wsh_501-700	predict	1	0	0.46	0.49	0	sig.
20220321__1115	Rural	_1118	5	41	Clr_Strp_Red__10W	block4_pool	0.98	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	41	Clr_Strp_Red__10W	block4_pool	0.98	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	41	Clr_Strp_Red__10W	block5_pool	0.99	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	41	Clr_Strp_Red__10W	fc1	0.99	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	41	Clr_Strp_Red__10W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	41	Clr_Strp_Red__10W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	42	Clr_Strp_Red__20W	block4_pool	0.98	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	42	Clr_Strp_Red__20W	block4_pool	0.98	0	0.48	0.45	0	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20220310__2139	Rural	_1118	5	42	Clr_Strp_Red__20W	block5_pool	0.99	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	42	Clr_Strp_Red__20W	fc1	0.99	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	42	Clr_Strp_Red__20W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	42	Clr_Strp_Red__20W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	43	Clr_Strp_Red__30W	block4_pool	0.76	0.37	0.52	0.45	0.01	sig.
20220310__2139	Rural	_1118	5	43	Clr_Strp_Red__30W	block4_pool	0.87	0.29	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	43	Clr_Strp_Red__30W	block5_pool	0.98	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	43	Clr_Strp_Red__30W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	43	Clr_Strp_Red__30W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	43	Clr_Strp_Red__30W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	44	Clr_Strp_Red__40W	block4_pool	0	0.02	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	44	Clr_Strp_Red__40W	block4_pool	0	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	44	Clr_Strp_Red__40W	block5_pool	0.98	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	44	Clr_Strp_Red__40W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	44	Clr_Strp_Red__40W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	44	Clr_Strp_Red__40W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	45	Clr_Strp_Ylw__10W	block4_pool	0.03	0.01	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	45	Clr_Strp_Ylw__10W	block4_pool	0.03	0.01	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	45	Clr_Strp_Ylw__10W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	45	Clr_Strp_Ylw__10W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	45	Clr_Strp_Ylw__10W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	45	Clr_Strp_Ylw__10W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	46	Clr_Strp_Ylw__20W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	46	Clr_Strp_Ylw__20W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	46	Clr_Strp_Ylw__20W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	46	Clr_Strp_Ylw__20W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	46	Clr_Strp_Ylw__20W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	46	Clr_Strp_Ylw__20W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	47	Clr_Strp_Ylw__30W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	47	Clr_Strp_Ylw__30W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	47	Clr_Strp_Ylw__30W	block5_pool	0.81	0.38	0.51	0.49	0.004	sig.
20220310__2139	Rural	_1118	5	47	Clr_Strp_Ylw__30W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	47	Clr_Strp_Ylw__30W	fc2	0.61	0.48	0.49	0.49	0.302	not sig.
20220310__2139	Rural	_1118	5	47	Clr_Strp_Ylw__30W	predict	0.95	0.21	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	48	Clr_Strp_Ylw__40W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	48	Clr_Strp_Ylw__40W	block4_pool	0.02	0.01	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	48	Clr_Strp_Ylw__40W	block5_pool	0.66	0.46	0.51	0.49	0.177	not sig.
20220310__2139	Rural	_1118	5	48	Clr_Strp_Ylw__40W	fc1	0.95	0.22	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	48	Clr_Strp_Ylw__40W	fc2	0.06	0.22	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	48	Clr_Strp_Ylw__40W	predict	0.16	0.35	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	49	Clr_Strp_Grn__10W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	49	Clr_Strp_Grn__10W	block4_pool	0.02	0	0.48	0.45	0	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20220310__2139	Rural	_1118	5	49	Clr_Strp_Grn__10W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	49	Clr_Strp_Grn__10W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	49	Clr_Strp_Grn__10W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	49	Clr_Strp_Grn__10W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	50	Clr_Strp_Grn__20W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	50	Clr_Strp_Grn__20W	block4_pool	0.02	0.01	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	50	Clr_Strp_Grn__20W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	50	Clr_Strp_Grn__20W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	50	Clr_Strp_Grn__20W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	50	Clr_Strp_Grn__20W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	51	Clr_Strp_Grn__30W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	51	Clr_Strp_Grn__30W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	51	Clr_Strp_Grn__30W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	51	Clr_Strp_Grn__30W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	51	Clr_Strp_Grn__30W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	51	Clr_Strp_Grn__30W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	52	Clr_Strp_Grn__40W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	52	Clr_Strp_Grn__40W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	52	Clr_Strp_Grn__40W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	52	Clr_Strp_Grn__40W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	52	Clr_Strp_Grn__40W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	52	Clr_Strp_Grn__40W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	53	Clr_Strp_BGn__10W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	53	Clr_Strp_BGn__10W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	53	Clr_Strp_BGn__10W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	53	Clr_Strp_BGn__10W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	53	Clr_Strp_BGn__10W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	53	Clr_Strp_BGn__10W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	54	Clr_Strp_BGn__20W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	54	Clr_Strp_BGn__20W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	54	Clr_Strp_BGn__20W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	54	Clr_Strp_BGn__20W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	54	Clr_Strp_BGn__20W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	54	Clr_Strp_BGn__20W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	55	Clr_Strp_BGn__30W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	55	Clr_Strp_BGn__30W	block4_pool	0.02	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	55	Clr_Strp_BGn__30W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	55	Clr_Strp_BGn__30W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	55	Clr_Strp_BGn__30W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	55	Clr_Strp_BGn__30W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	56	Clr_Strp_BGn__40W	block4_pool	0.02	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	56	Clr_Strp_BGn__40W	block4_pool	0.02	0	0.48	0.45	0	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20220310__2139	Rural	_1118	5	56	Clr_Strp_BGn__40W	block5_pool	1	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	56	Clr_Strp_BGn__40W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	56	Clr_Strp_BGn__40W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	56	Clr_Strp_BGn__40W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	57	Clr_Strp_Blu__10W	block4_pool	0.98	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	57	Clr_Strp_Blu__10W	block4_pool	0.98	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	57	Clr_Strp_Blu__10W	block5_pool	0.99	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	57	Clr_Strp_Blu__10W	fc1	0.99	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	57	Clr_Strp_Blu__10W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	57	Clr_Strp_Blu__10W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	58	Clr_Strp_Blu__20W	block4_pool	0.98	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	58	Clr_Strp_Blu__20W	block4_pool	0.98	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	58	Clr_Strp_Blu__20W	block5_pool	0.99	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	58	Clr_Strp_Blu__20W	fc1	0.99	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	58	Clr_Strp_Blu__20W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	58	Clr_Strp_Blu__20W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	59	Clr_Strp_Blu__30W	block4_pool	0.92	0.21	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	59	Clr_Strp_Blu__30W	block4_pool	0.97	0.01	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	59	Clr_Strp_Blu__30W	block5_pool	0.98	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	59	Clr_Strp_Blu__30W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	59	Clr_Strp_Blu__30W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	59	Clr_Strp_Blu__30W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	60	Clr_Strp_Blu__40W	block4_pool	0.04	0.16	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	60	Clr_Strp_Blu__40W	block4_pool	0	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	60	Clr_Strp_Blu__40W	block5_pool	0.98	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	60	Clr_Strp_Blu__40W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	60	Clr_Strp_Blu__40W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	60	Clr_Strp_Blu__40W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	61	Clr_Strp_Vlt__10W	block4_pool	0.98	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	61	Clr_Strp_Vlt__10W	block4_pool	0.98	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	61	Clr_Strp_Vlt__10W	block5_pool	0.99	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	61	Clr_Strp_Vlt__10W	fc1	0.99	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	61	Clr_Strp_Vlt__10W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	61	Clr_Strp_Vlt__10W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	62	Clr_Strp_Vlt__20W	block4_pool	0.98	0	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	62	Clr_Strp_Vlt__20W	block4_pool	0.98	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	62	Clr_Strp_Vlt__20W	block5_pool	0.99	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	62	Clr_Strp_Vlt__20W	fc1	0.99	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	62	Clr_Strp_Vlt__20W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	62	Clr_Strp_Vlt__20W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	63	Clr_Strp_Vlt__30W	block4_pool	0.97	0.01	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	63	Clr_Strp_Vlt__30W	block4_pool	0.98	0.01	0.48	0.45	0	sig.

TCAV Test Record	Class	Internal VGG16 Version	Thesis VGG16 #	Concept #	TCAV Concept	VGG16 Bottleneck	TCAV Score	Score Error (+/-)	Random TCAV Score	ran error (+/-)	p-val	Sig. / not sig.
20220310__2139	Rural	_1118	5	63	Clr_Strp_Vlt__30W	block5_pool	0.98	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	63	Clr_Strp_Vlt__30W	fc1	0.95	0.22	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	63	Clr_Strp_Vlt__30W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	63	Clr_Strp_Vlt__30W	predict	1	0	0.51	0.5	0	sig.
20220321__1115	Rural	_1118	5	64	Clr_Strp_Vlt__40W	block4_pool	0.01	0.05	0.52	0.45	0	sig.
20220310__2139	Rural	_1118	5	64	Clr_Strp_Vlt__40W	block4_pool	0	0	0.48	0.45	0	sig.
20220310__2139	Rural	_1118	5	64	Clr_Strp_Vlt__40W	block5_pool	0.98	0	0.51	0.49	0	sig.
20220310__2139	Rural	_1118	5	64	Clr_Strp_Vlt__40W	fc1	1	0	0.47	0.5	0	sig.
20220310__2139	Rural	_1118	5	64	Clr_Strp_Vlt__40W	fc2	1	0	0.49	0.49	0	sig.
20220310__2139	Rural	_1118	5	64	Clr_Strp_Vlt__40W	predict	1	0	0.51	0.5	0	sig.
20220418__2107	Rural	_1118	5	65	Arcs_040_K010__Red	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	66	Arcs_040_K010__Ylw	predict	0.02	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	67	Arcs_040_K010__Grn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	68	Arcs_040_K010__BGn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	69	Arcs_040_K010__Blu	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	70	Arcs_040_K010__Vlt	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	71	Arcs_040_K030__Red	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	72	Arcs_040_K030__Ylw	predict	0.31	0.45	0.48	0.5	0.136	not sig.
20220418__2107	Rural	_1118	5	73	Arcs_040_K030__Grn	predict	0.95	0.21	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	74	Arcs_040_K030__BGn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	75	Arcs_040_K030__Blu	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	76	Arcs_040_K030__Vlt	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	77	Arcs_040_K070__Red	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	78	Arcs_040_K070__Ylw	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	79	Arcs_040_K070__Grn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	80	Arcs_040_K070__BGn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	81	Arcs_040_K070__Blu	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	82	Arcs_040_K070__Vlt	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	83	Arcs_040_K100__Red	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	84	Arcs_040_K100__Ylw	predict	0.85	0.35	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	85	Arcs_040_K100_Grn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	86	Arcs_040_K100__BGn	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	87	Arcs_040_K100__Blu	predict	1	0	0.48	0.5	0	sig.
20220418__2107	Rural	_1118	5	88	Arcs_040_K100__Vlt	predict	1	0	0.48	0.5	0	sig.

REFERENCES

Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." arXiv preprint arXiv:1605.07678 (2016).

Chiang, Cheng-Han, and Hung-yi Lee. "Pre-Training a Language Model Without Human Language." arXiv preprint arXiv:2012.11995 (2020).

Clough, James R., Ilkay Oksuz, Esther Puyol-Antón, Bram Ruijsink, Andrew P. King, and Julia A. Schnabel. "Global and local interpretability for cardiac MRI classification." In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 656-664. Springer, Cham, 2019.

Denninger, Maximilian, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomas Hodan, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi. "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering." In Robotics: Science and Systems (RSS) Workshops, vol. 2, no. 3, p. 7. 2020.

Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An open urban driving simulator." arXiv preprint arXiv:1711.03938 (2017).

Felbo, Bjarke, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm." arXiv preprint arXiv:1708.00524 (2017).

Ghorbani, Amirata, James Wexler, James Zou, and Been Kim. "Towards automatic concept-based explanations." arXiv preprint arXiv:1902.03129 (2019).

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249-256. JMLR Workshop and Conference Proceedings, 2010.

Hinterstoisser, Stefan, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. "On pre-trained image features and synthetic images for deep learning." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 0-0. 2018.

Hinterstoisser, Stefan, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. "On pre-trained image features and synthetic images for deep learning." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 0-0. 2018.

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012).

Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." arXiv preprint arXiv:1801.06146 (2018).

Johnson-Roberson, Matthew, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?." arXiv preprint arXiv:1610.01983 (2016).

Kim, Been, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, and Fernanda Viegas. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)." In International conference on machine learning, pp. 2668-2677. PMLR, 2018.

Linder, Timm, MJ Hernandez Leon, Narunas Vaskevicius, and Kai O. Arras. "Towards training person detectors for mobile robots using synthetically generated RGB-D data." In Computer Vision and Pattern Recognition (CVPR) 2019 Workshop on 3D Scene Generation. 2019.

Liu, Tony, and Arvid Mildner. "Training Deep Neural Networks on Synthetic Data." LU-CS-EX (2020).

Liu, Weixing, Jun Liu, and Bin Luo. "Can Synthetic Data Improve Object Detection Results for Remote Sensing Images?." arXiv preprint arXiv:2006.05015 (2020).

Mincu, Diana, Eric Loreaux, Shaobo Hou, Sebastien Baur, Ivan Protsyuk, Martin Seneviratne, Anne Mottram, Nenad Tomasev, Alan Karthikesalingam, and Jessica Schrouff. "Concept-based model explanations for Electronic Health Records." In Proceedings of the Conference on Health, Inference, and Learning, pp. 36-46. 2021.

Neyshabur, Behnam, Hanie Sedghi, and Chiyuan Zhang. "What is being transferred in transfer learning?." arXiv preprint arXiv:2008.11687 (2020).

Nowruzi, Farzan Erlik, Prince Kapoor, Dhanvin Kolhatkar, Fahed Al Hassanat, Robert Laganiere, and Julien Rebut. "How much real data do we actually need: Analyzing object detection performance using synthetic and real data." arXiv preprint arXiv:1907.07061 (2019).

Prakash, Aayush, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. "Structured domain randomization: Bridging the reality gap by context-aware synthetic data." In 2019 International Conference on Robotics and Automation (ICRA), pp. 7249-7255. IEEE, 2019.

Pujitha Appan, K., and Jayanthi Sivaswamy. "Retinal Image Synthesis for CAD development." (2018).

Raghu, Maithra, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. "Transfusion: Understanding transfer learning for medical imaging." Advances in neural information processing systems 32 (2019).

Rajpura, Param S., Hristo Bojinov, and Ravi S. Hegde. "Object detection using deep CNNs trained on synthetic images." arXiv preprint arXiv:1706.06782 (2017).

Reiersen, Magnus. "Deep Visual Domain Adaptation:-From Synthetic Data to the Real World." Master's thesis, NTNU, 2018.

Reyes, Mauricio, Raphael Meier, Sérgio Pereira, Carlos A. Silva, Fried-Michael Dahlweid, Hendrik von Tengg-Kobligk, Ronald M. Summers, and Roland Wiest. "On the interpretability of artificial intelligence in Radiology: Challenges and Opportunities." Radiology: Artificial Intelligence 2, no. 3 (2020): e190043.

Ros, German, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3234-3243. 2016.

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

Soni, Rahul, Naresh Shah, Chua Tat Seng, and Jimmy D. Moore. "Adversarial TCAV--Robust and Effective Interpretation of Intermediate Layers in Neural Networks." arXiv preprint arXiv:2002.03549 (2020).

Sprague, Conner, Eric B. Wendoloski, and Ingrid Guch. "Interpretable AI for Deep Learning− Based Meteorological Applications." In 99th American Meteorological Society Annual Meeting. AMS, 2019.

Tremblay, Jonathan, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 969-977. 2018.

White, Gary, Christian Cabrera, Andrei Palade, Fan Li, and Siobhan Clarke. "WasteNet: Waste classification at the edge for smart bins." arXiv preprint arXiv:2006.05873 (2020).

Wiedemann, Gregor, Eugen Ruppert, Raghav Jindal, and Chris Biemann. "Transfer learning from lda to bilstm-cnn for offensive language detection in twitter." arXiv preprint arXiv:1811.02906 (2018).

Wrenninge, Magnus, and Jonas Unger. "Synscapes: A photorealistic synthetic dataset for street scene parsing." arXiv preprint arXiv:1810.08705 (2018).

Yeh, Chih-Kuan, Been Kim, Sercan O. Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. "On completeness-aware concept-based explanations in deep neural networks." arXiv preprint arXiv:1910.07969 (2019).

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?." Advances in neural information processing systems 27 (2014).

BIOGRAPHY

Brian L Shaw received his Bachelor of Science from Virginia Polytechnic Institute and has worked as a materials scientist and management consultant among other responsibilities during past few decades.