# Technical Report: Evaluation of Transfer Learning for Viewpoint Classification

**Jason Parham**
Department of Computer Science
Rensselaer Polytechnic Institute
parhaj@rpi.edu

**Zachary Jablons**
Department of Computer Science
Rensselaer Polytechnic Institute
jabloz2@rpi.edu

**Charles V. Stewart**
Department of Computer Science
Rensselaer Polytechnic Institute
stewart@cs.rpi.edu

## Abstract

In this report we explore the use of transfer learning for Convolutional Neural Networks (CNN) in order to apply networks trained for the popular ILSVRC challenge to a viewpoint classification problem. In the viewpoint classification task, the network must learn to distinguish not just between different species of animals, but also between the 8 cardinal and ordinal viewpoints for each species of animal. We transfered the learning from the pre-trained *accurate* OverFeat network architecture, provided by NYU, to achieve a viewpoint classification accuracy of 80.20 % for grid sampled patches and 87.93 % for animals with pre-defined bounding boxes.

## 1 Introduction

Due to the nature of convolutional neural networks, it is possible to train over one dataset and - in a word - transfer the learning to a new network in order to perform on a different dataset and potentially even a different task altogether. In this application, the transferred learning can be treated as pre-training for the new neural network's architecture and the weights would be further fine-tuned to specialize to the new task [**?**]. Transfer learning gives the advantage of not only allowing for custom structures to fit different types of datasets, but also lets researchers take advantage of the convolutional filters learned on larger, more complex corpora. This results with a more robust model in less training time compared to starting from randomly initialized weights and can also help augment datasets that are too small to to train a full CNN on their own.

In light of the recent successes of convolutional neural networks on large-scale image classification problems, some researchers have also opted to treat these models as black-box like feature extractors, on top of which other machine learning algorithms can be trained [**?**]. Additionally, recent experiments [**?**] have shown through visualizations of the lower layers of a convolutional neural network, that the lower layers tend to learn features salient for extracting object information regardless of class, becoming more specialized deeper in the network. For a more thorough survey of transfer learning and the visual properties of each level of convolutional filters, we refer the reader to the analysis by Yosinski *et al.* [**?**].

The OverFeat network [**?**] is one such pre-trained convolutional neural network, which was trained on the data provided in the ILSVRC2013 competition [**?**] and consequently won the localization task in the competition. This network's architecture and the weights used during the competition are provided by Sermanet *et al.* and is well supported by popular neural network libraries - making it an ideal candidate for transferring

learned features. Our analysis and the resulting networks we train will be utilizing the convolutional layers of the *accurate* OverFeat model as the bottom layers of our network, as detailed in Section 4.1.

The rest of the paper is as follows. Section 2 details the recent work in the computer vision field on convolutional neural networks, transfer learning, and viewpoint estimation. Section 3 details the dataset we used to train our networks and the collection techniques. Section 4 details the different models we trained for our viewpoint classification task. Section 5 details the experimental results. Section 6 details an explanation of the results and a breif overview of future work.

## 2 Recent Work

Convolutional neural networks have become increasingly popular for computer vision tasks. The learned kernel convolutions of a neural network are applied to an image in a translation-invariant fashion to produce a sophisticated recognition task. When trained with back-propagation and gradient descent, they were first used to great success by LeCun *et al.* [**?**] on the MNIST digit recognition task in 1998. More recently, due in part to the Dropout regularization method [**?**], convolutional neural networks have been very successful in object classification with large amounts of classes, like in the ILSVRC competition [**?**, **?**].

Due to the successes of transfer learning, the dataset on which an original network was trained on becomes very important. The ImageNet ILSVRC2013 challenge provides one such dataset, containing 1.2 million images labeled with 1000 different classes. These are images of real world objects, including natural and man-made objects. Thus, this dataset provides a good basis for training a convolutional network for transfer learning into tasks involving images of real world objects, as many of the lower level features learned by the network can be reused.

As in [**?**], the varying viewpoints can be treated as separate parts in a deformable parts model [**?**]. For datasets that are not so detailed in their ground-truth, explicit exemplars of each viewpoint [**?**] can be used to estimate pose. An alternative approach for viewpoint estimation and automatic classification with neural networks was detailed by Yang *et al.* [**?**]; furthermore, a more detailed survey of pose estimation with neural networks can be reviewed in this paper.

## 3 Dataset

African Elephant | Giraffe | Grevy's Zebra | Plains Zebra

Figure 1: exemplars for each of the 4 species we trained our networks on

### 3.1 Source Images

The networks trained in this paper are trained from a dataset of 3,817 images containing animals enclosed with bounding boxes and annotated with species and viewpoint (e.g. elephant front) information saved in the PASCAL [**?**] annotation format. The images contain the 4 following species: African Elephants, Giraffes, Grevy's Zebras, and Plains Zebras. Example images of each of these species can be found in Figure 3. The viewpoint of each species is determined relative to the camera, describing essentially what part of the animal is being seen. The viewpoints of each species is partitioned into eight bins (cardinal Front, Back, Left, Right and ordinal Front Left, Front Right, Back Left, and Back Right) resulting in 32 viewpoint classes plus an additional negative class. The analysis of the transfered OverFeat learning is divided into two datasets extracted from this image dataset: square patches sampled on a grid across the image and the resized

bounding boxes around each animal in the annotated dataset. It is important to mention that Giraffe and Grevy's Zebra images are not provided in the ILSVRC2013 competition dataset, whereas African Elephant and Plains Zebra images are provided.

### 3.2 Annotations

To extract the annotation image patches, we simply cropped the bounding boxes of each animal and naively resize the resulting image to 221x221 pixels, without preserving the original aspect ratio - this is the similar technique utilized in [**?**] to prepare images for training. We will refer to these extracted images as the *annotations dataset*[1], and does not include negative patches. A total of 5,304 annotations were extracted from the source images.

### 3.3 Patches

The grid patches were extracted from a 221x221 pixel sliding window throughout the image, where the label of each window was assigned by what label overlapped with the most area (giving a negative class for not overlapping with over 50% of some bounding box's area). We will refer to these extracted images as the *patches dataset*. Due to the dense sampling, the patches dataset provides far more images to train with than the annotations dataset. One major issue with the patches dataset is that it creates issues of scale because we did not perform any size normalization, which cannot realistically be done without a pixel-level segmentation of the image. The resulting scale problem results in patches that are extremely difficult to classify - even for a human evaluator. A total of 122,854 patches were extracted; sample patches can be viewed in the Appendix along with the class distribution of the patches can be viewed in Table 7.

## 4 Models

### 4.1 OverFeat

OverFeat is a neural network pre-trained for the ILSVRC2013 classification, localization, and detection tasks. There are two versions available: a *fast* model and an *accurate* model. Since we did not notice a significant difference in speed between these two versions on our Theano GPU implementation, we opted for the *accurate* model to utilize more feature dimensions. In order to transfer the learned weights from OverFeat to our other networks, we construct our networks by starting after the convolutional layers of OverFeat and fixing the weights. For our experimental purposes, this results in making OverFeat a black-box feature extractor. The *accurate* model takes fixed-size input as a 221x221 pixel, 3-channel color image, whereas the *fast* model takes as input a 231x231 pixel, 3-channel image. For each dataset, we created two sets of features based on the output activations of different layers of OverFeat.

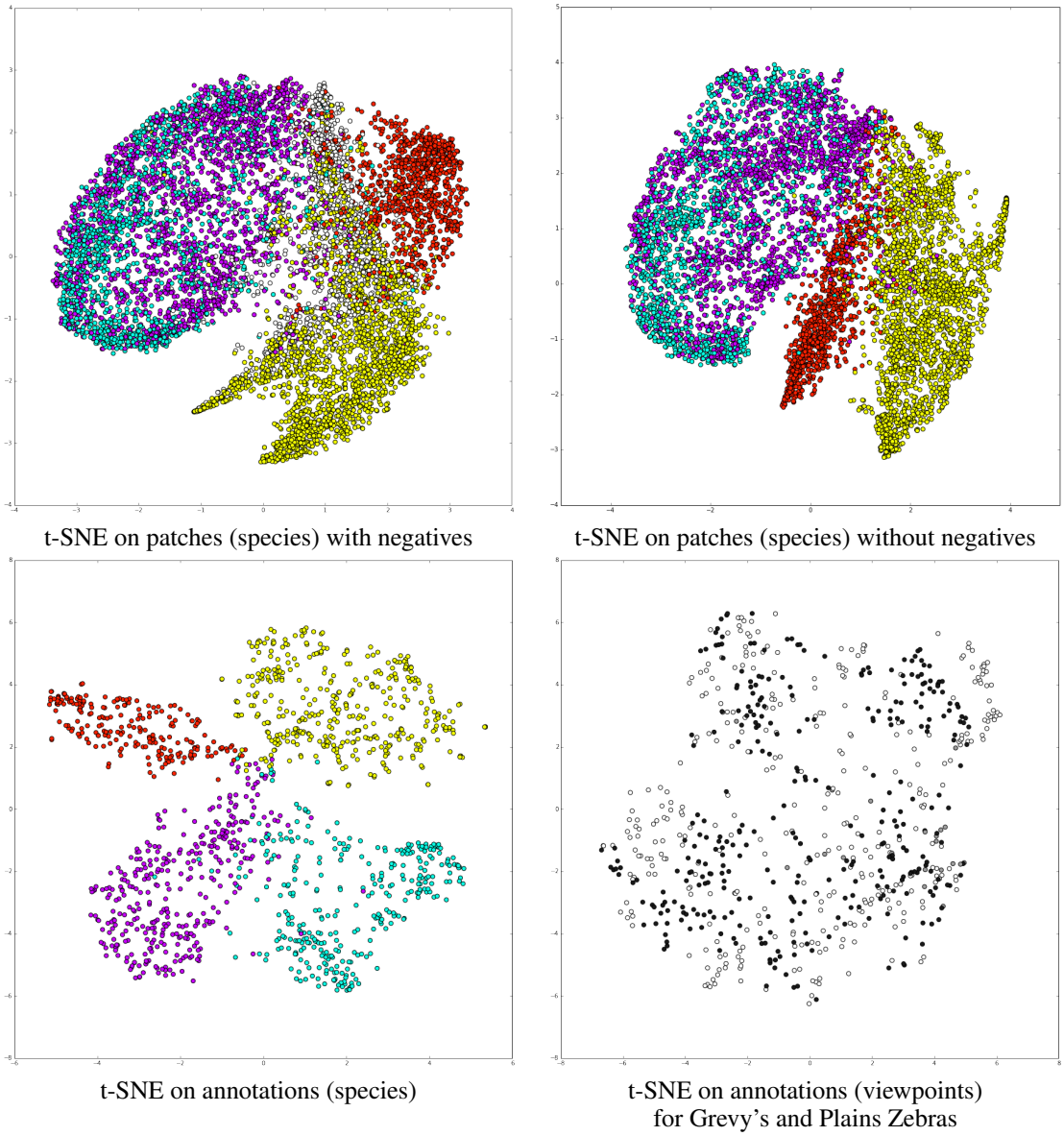
1. *flat* features - the activations from the first non-convolutional, fully-conencted layer of OverFeat (layer 21), which outputs a flat 4096-dimensional vector.
2. *square* features - the activations from the last convolutional layer of OverFeat (layer 19), which gives us tensors consisting of 1024-channel, 5x5 matrices.

Since OverFeat uses rectified linear activations, these features are around 80% to 90% sparse. For more details on the specific structure of the network, we refer the reader to Sermanet *et al.*'s paper [**?**] and Krizhevsky *et al.*'s paper [**?**].

In order to qualitatively assess how well separated the data obtained from OverFeat was by our classes, we computed a 2-dimensional t-SNE [**?**] embedding of the *flat* features computed on the *patches dataset*, shown in Figure 4.1 and computer by scikit-learn's t-SNE implementation [**?**]. As negatives can oftentimes contain large parts of an animal and its background, we are unsurprised there is a large amount of confusion between the species labels. However, we can see that Giraffes, a class that does not exist in ImageNet, is separated very nicely from the rest of the classes despite OverFeat having never seen that class before. Zebras are

[1]Networks and algorithms with the subscript $A$ were trained using the annotation training data instead of the patches training data.

unsurprisingly mixed in their own cluster, which is likely due to both the images being very similar between species[2] and OverFeat being trained to only recognize Plains Zebras.



t-SNE on patches (species) with negatives

t-SNE on patches (species) without negatives

t-SNE on annotations (species)

t-SNE on annotations (viewpoints) for Grevy's and Plains Zebras

Figure 2: (Species) Red: African Elephant, Yellow: Giraffe, Blue: Zebra Grevy's, Purple: Zebra Plains, White: Negative; (Viewpoints) Black: Left-side [Front Left, Left, Back Left], White: Right-side [Front Right, Right, Back Right], Grey: Front and Back

The t-SNE visualization was also run on the *annotations dataset*, which has OverFeat patches extracted on resized, pre-defined bounding boxes. We can see immediately a strikingly clear separation that OverFeat provides and does an almost perfect job at separating and clustering the images into the correct species. This result is not surprising considering that the general convolutional filters from OverFeat are able to capture macro-level details about the animals and the learning can be effectively transferred to classes that were not trained on. Interestingly, OverFeat cannot effectively separate the left and right-size viewpoints for Grevy's

[2]In fact, we had some issues with human annotators mixing up Grevy's versus Plains Zebras during data collection

and Plains Zebras for the t-SNE visualization shows much more confusion. This is where the specialization and fine-tuning of the OverFeat fully-conencted layers can help separate the viewpoints more accurately.

Other than t-SNE, we did attempt to train a generative, autoencoder model in order to visualize the separation of the OverFeat feature vectors. Ultimately, due to restrictions in time and limitations of the Caffe implementation, we were unable to train an effective network. We attribute our difficulties to an inadequate weight initialization scheme and the network being unable to regenerate the such sparse input vectors that OverFeat provides.

### 4.2 Architectures

On top of the *flat* features, we trained the following four architectures on the *patches dataset*. All hidden units were rectified linear units except for the softmax layer, and all models were implemented in Caffe [**?**].

- **Linear**: A linear model (i.e. a softmax layer connected directly - with weights - to the data)
- **H1**: A network with a single hidden layer of 256 units
- **H2**: A network with two hidden layers: 1024 units and then 256 units.
- **H4**: A network with four hidden layers of size 2048, 1024, 512, 256

### 4.3 Parameter Search

Neural networks have a reputation for being notoriously difficult to train because of the relatively large number of hyper-parameters; having many hyper-parameters to tune results in having to search a very large, high-dimensional parameter space and - especially for neural networks - testing a particular configuration takes non-trivial computational and time resources. As described in [**?**], the search for optimal hyper-parameters for a given network architecture can be by a manual, grid, or random protocol. If a random search is chosen, a Bayesian learning algorithm can be used to help evaluate which dimensions and ranges of configurations offer better accuracy. In order to limit the size of the search space and reduce the complexity of our overall architectures, we limited the number of hyper-parameters to 5, which are detailed in Table 4.3.

| **Hyper-Parameter** | **Description** |
|---|---|
| Learning Rate | Scale each weight gradient by *Learning Rate* |
| Gamma | After each *Step Size*, multiply (decrease) the *Learning Rate* by *Gamma* |
| Step Size | After how many iterations to decrease the *Learning Rate* by *Gamma* |
| Momentum | Averave the current gradient with the previous gradient by *Momentum* |
| Batch Size | How many training examples to use in each mini-batch |

Table 1: hyper-parameters and their descriptions

For the 4 network architectures that were trained, we randomly searched for optimal hyper-parameters for each network independently. The final hyper-parameters chosen for each network can be found in Table 7 in the Appendix.

## 5 Results

### 5.1 Patches

In order to establish a baseline for this task, we also trained an SVM with an RBF kernel ($\gamma = \frac{1}{4096}$) on 25% of the data (due to time limitations). We also ran a $k$-nearest-neighbor algorithm with $k = 33$. The results are given in Table 7, with the subscript A for the models trained on the annotations dataset. Although the training and testing sets for the SVM were significantly smaller than the neural networks, the performance difference between it and the simple **Linear** model discourage further exploration of this method. The k-NN is markedly worse than either SVM or neural network, which is unsurprising.

Between the networks, it is clear that adding a hidden layer greatly improves accuracy, but that adding too many hidden layers has diminishing returns, and for the *patches dataset* the network starts overfitting as more layers are introduced and adequate regularization becomes increasingly elusive. The two layer network appears to perform the best, and it mimics the number of layers that come after the extracted one in OverFeat.

| **Algorithm** | **33-class Accuracy** | **Species Accuracy** | **Viewpoint Accuracy** |
|---|---|---|---|
| **k-NN** | 42.86 % | 73.55 % | 49.97 % |
| **SVM** | 45.64 % | 81.26 % | 54.13 % |
| **Linear** | 54.13 % | 87.42 % | 73.90 % |
| **H1** | 63.43 % | 90.07 % | 78.33 % |
| **H2** | **66.14 %** | **91.42 %** | **80.20 %** |
| **H4** | 65.73 % | **91.46 %** | 74.99 % |
| **Algorithm** | **32-class Accuracy** | **Species Accuracy** | **Viewpoint Accuracy** |
| **k-NN**$_A$ | 25.39 % | 33.33 % | 25.52 % |
| **SVM**$_A$ | 18.30 % | 32.86 % | 22.95 % |
| **Linear**$_A$ | 70.74 % | 97.66 % | 86.88 % |
| **H1**$_A$ | 71.57 % | 97.66 % | 87.71 % |
| **H2**$_A$ | **72.17** % | 97.81 % | 87.71 % |
| **H4**$_A$ | 70.97 % | **98.11** % | **87.93** % |

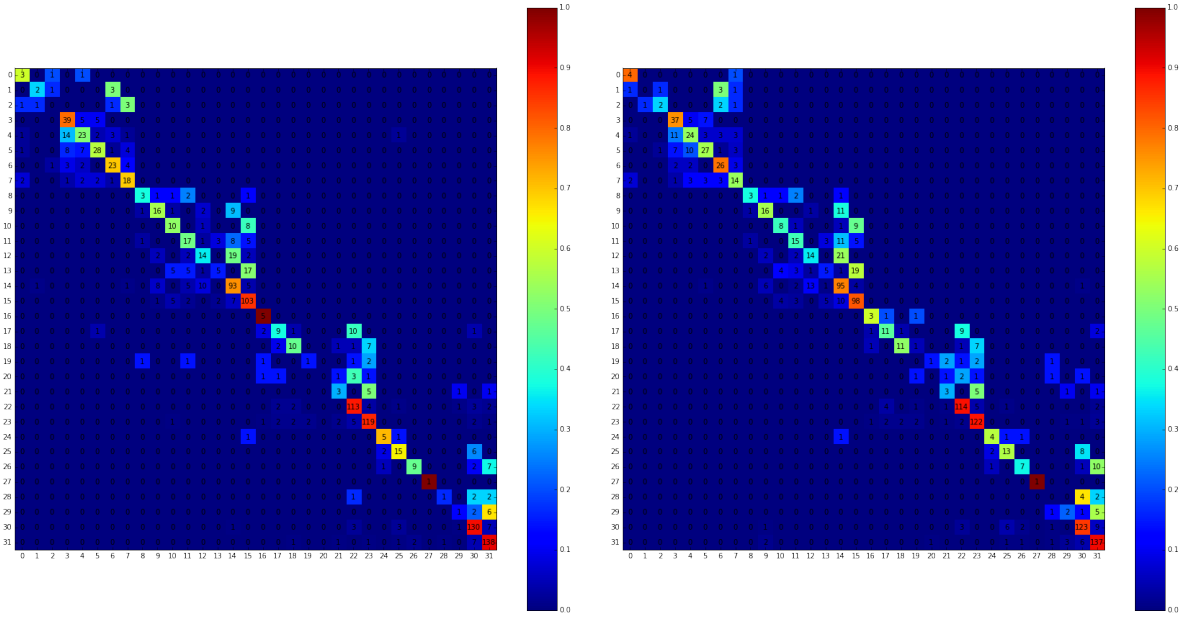
Table 2: hyper-parameters used to train each architecture

A closer inspection of the confusion matrices given in Figure 7 in the appendix shows similar patterns in confusion. Notably, the negatives (label 16) are often confused for non-negative labels, which is unsurprising given that they may still contain up to 50% of the bounding box of a labeled animal. Another pattern noticeable is the confusion between viewpoint for particular species. Elephants seem to suffer from this the least, however Giraffes and Zebras both end up having the Left and Right viewpoints confused. This can be explained by the fact that the 4096-dimensional feature vector doesn't contain explicit spatial information that could be used to 'undo' the transposition invariance learned by OverFeat on ImageNet (i.e. a Zebra in ImageNet is given the same label regardless of whether its facing left or right). However, since a Zebra rump looks very different from, for example, its flank, the network seems to be able to learn the distinction and identify those viewpoints. It may also be due to the fact that patches sampled from the right flank of a zebra do not look very different from those sampled from the left flank at a small scale. While there seems to be a similar pattern for Giraffes and Elephants, it is not as drastic. Additionally, there is significantly more species confusion between Plains and Grevy's Zebras than between any other two species, which may be a result of the OverFeat network being trained to recognize the general class 'Zebra' rather than any finer distinction.

As can be seen in Table 7, the SVM and k-NN baselines perform much worse on this smaller dataset than the neural networks, which actually perform better than on the *patches dataset*. We note that since there are no negative images in the *annotations dataset*, the lack of aforementioned confusion between a negative and anything else due to bounding box overlap is not a concern, and may give the performance boost seen here. However, this does mean that the networks learned may not be capable of providing good detection capabilities later on. Additionally, in terms of the viewpoint, there are no ambiguous patches like many of those in Table 7 that even a human would have trouble distinguishing viewpoint from. With all these in mind, it is more surprising that the SVM and k-NN models did not do nearly as well as they did on the *patches dataset*.

The confusion matrices for the annotations, which can be seen in Figure 3, show far less confusion in general, especially between left and right viewpoints of zebras. This gives greater weight to the idea that the patches may be indistinct between right and left viewpoints rather than the whole image of the animal. Regardless, there is still significant confusion between viewpoints in general, though not nearly as much as with the patches. This may also be due to the lack of a negative class - although there is a softmax unit corresponding to that class.

### 5.2 Annotations



**H2**$_A$ **H4**$_A$

Figure 3: confusion matrices for select annotations network architectures

## 6 Discussion and Future Work

Given the extreme class imbalance within the dataset between viewpoints within species, all trained networks will have performance issues distinguishing between certain viewpoints. If better training data is made available in the future, the networks can be retrained to potentially do a better job at discriminating between difficult viewpoints. Furthermore, adding artificial noise could help augment the dataset to correct for class imbalance, although these techniques are not evaluated in this paper.

In the immediate future, it would make sense to first re-run these experiments with the *square features*, and then secondly retrain the entire network (i.e. including every layer before the feature extraction) using the OverFeat weights for initialization rather than effectively fixing the weights up until the point of feature extraction. This would allow the network to retrain the lower level features to extract information relevant to viewpoint as well, at the cost of longer training time. This process is known as 'fine-tuning', and has been shown in [**?**] to be better overall on various splits of the ILSVRC task than holding the weights fixed. Since they only do object classification in [**?**], we propose that we might even get better results from fine-tuning.

The failed attempt to train an autoencoder from the sparse OverFeat feature vectors also is an area for improvement. An effective training of the network could be accomplished by densely encoding the input for more faithful regeneration; a more involved pre-training approach to initializing the autoencoder can also be implemented where each layer is trained one at a time as a deep belief network (followed by fine-tuning the entire stacked autoencoder network through the use of contrastive divergence [**?**]). This layer-by-layer approach, outlined in [**?**] by Hinton, could yield better visualization results grouped by salient features for image reconstruction, which may prove better clustered by viewpoint.

The next step from a patch-based convolutional neural network (like OverFeat) is to convolve the network architecture over the entire image and apply the entire pipeline as a convolution, as detailed in [**?**]. This fully convolutional approach to image segmentation can be seen as an extremely dense patch sampling and results in a pixel-level classification map across the entire image, while allowing transfer of a network learned for pure patch classification to this new task. Thus, the results in this paper should extend to such a network structure, transferring the learning of a viewpoint-centric view of the world.

## 7 Acknowledgements

The authors would like to thank the following: Mr. Kyle Kastner for the sklearn-theano package [**?**], which efficiently computed the OverFeat feature vectors on an NVIDIA GeForce GTX 660 GPU using Theano [**?**, **?**]; Mr. Patrick Wieschollеck for providing repository code to shuffle HDF5 data efficiently in Caffe [**?**]; the 11 RPI students - Messrs. Chen, Kamar, Machado, Poma, Raghuraman, Sanchez, Zielinski, Zhu, Misses Garcia, Tambling, and Wang - who worked on labeling and annotating the images for our dataset; and Mrs. Pam Paslow for helping to administer the data collection.

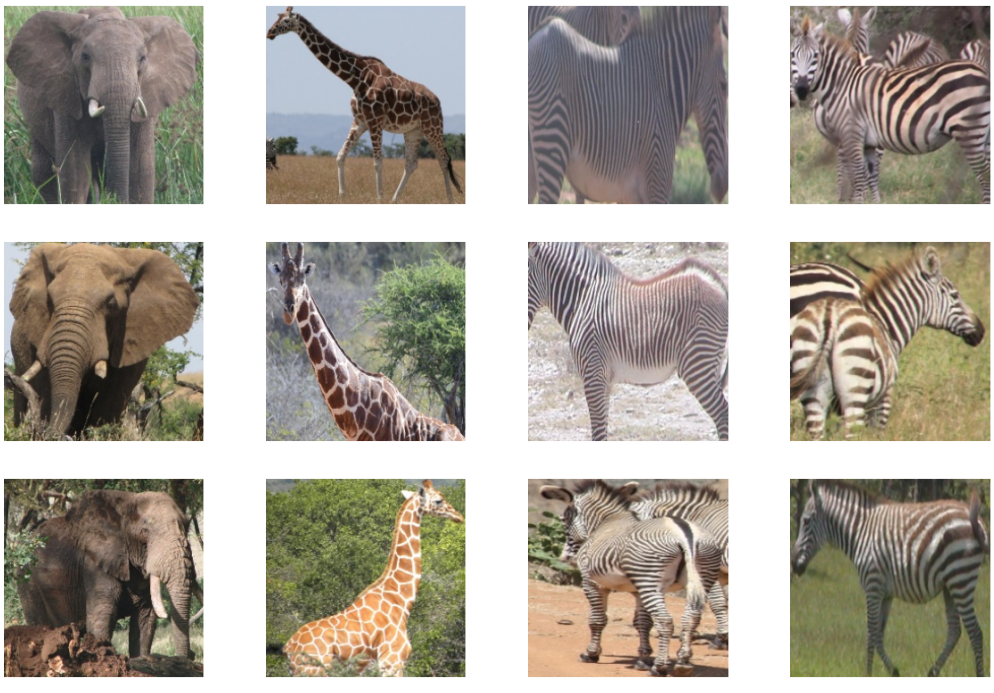
## Appendix

| Hyper-Parameter | Range Begin | Range End | Range Stride | Total Values |
|---|---|---|---|---|
| Learning Rate | 0.005 | 0.100 | 0.005 | 20 |
| Gamma | 0.10 | 0.60 | 0.05 | 11 |
| Step Size | 1000 | 15000 | 1000 | 15 |
| Momentum | 0.85 | 0.95 | 0.01 | 11 |
| Batch Size | 64 | 256 | 32 | 7 |
| **Total Configurations** | | | | 254,100 |

Table 3: hyper-parameter search dimensions

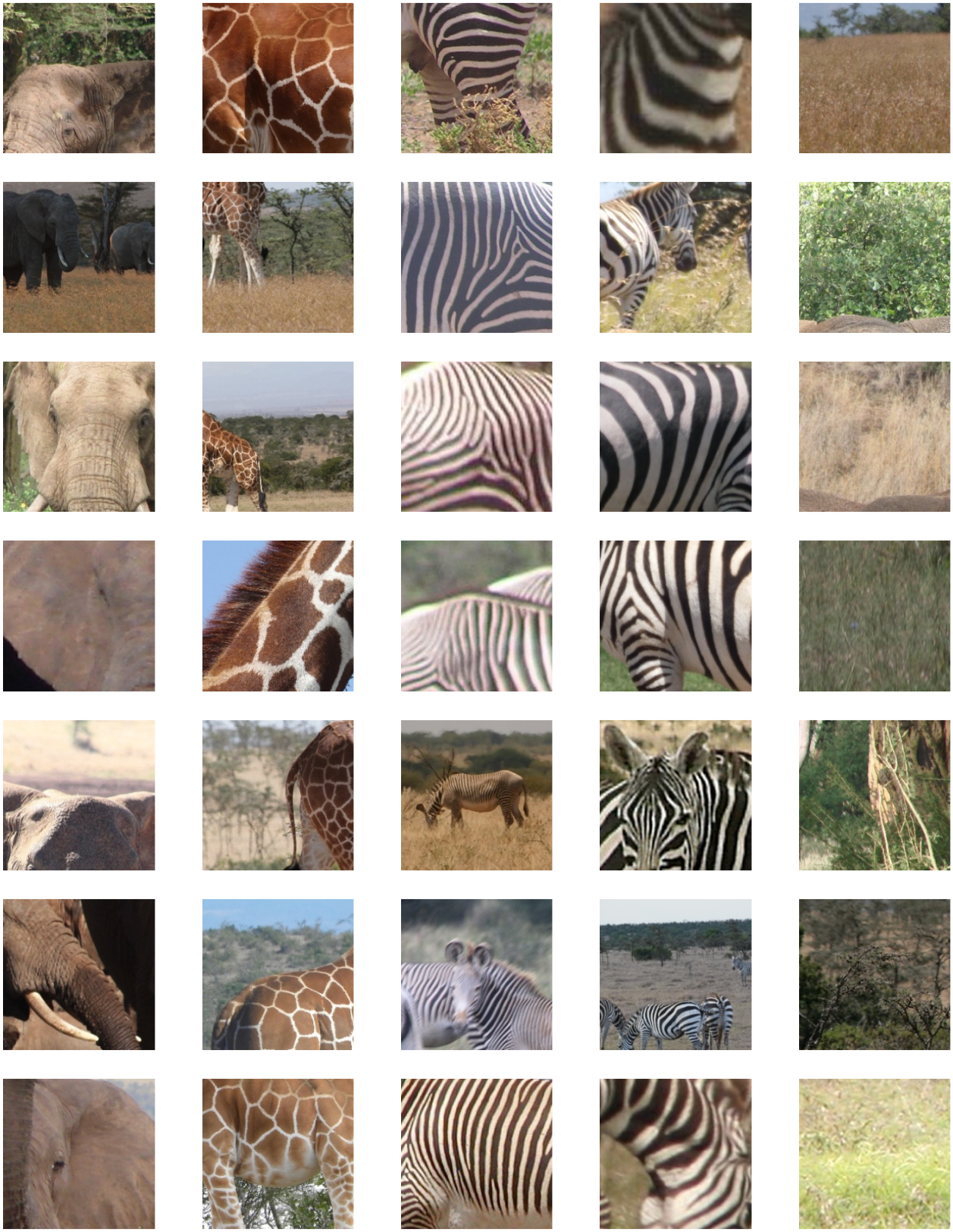
| Architecture | Learning Rate | Gamma | Step Size | Momentum | Batch Size |
|---|---|---|---|---|---|
| **Linear / Linear$_A$** | 0.050 | 0.35 | 6,000 | 0.89 | 128 |
| **H1 / H2$_A$** | 0.035 | 0.50 | 10,000 | 0.88 | 160 |
| **H2 / H2$_A$** | 0.045 | 0.45 | 10,000 | 0.88 | 192 |
| **H4** | 0.060 | 0.40 | 10,000 | 0.89 | 128 |
| **H4$_A$** | 0.050 | 0.30 | 10,000 | 0.90 | 192 |

Table 4: hyper-parameters used to train each architecture



African Elephant Giraffe Grevy's Zebra Plains Zebra

Table 5: sample annotation patches (no negatives) used during training **H4$_A$**

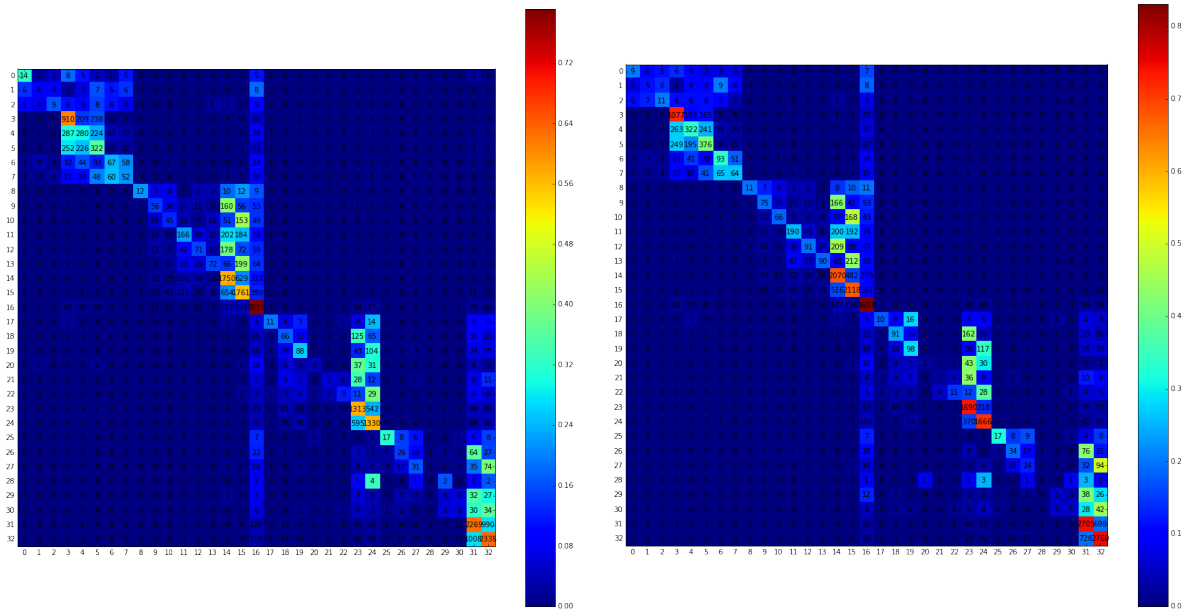


African Elephant | Giraffe | Grevy's Zebra | Plains Zebra | Negative

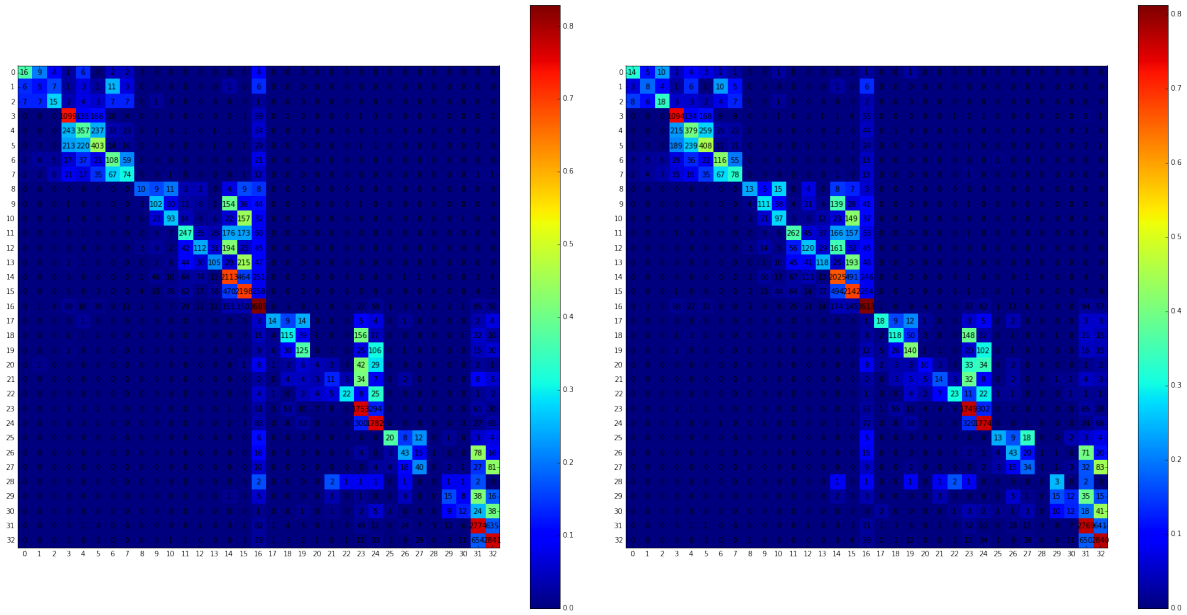
Table 6: sample patches used during training **Linear**, **H1**, **H2**, and **H4**

| Class Label | Species | Viewpoint | Total Patches | Percent of Dataset |
|---|---|---|---|---|
| 0 | African Elephant | Back | 168 | 0.137 % |
| 1 | African Elephant | Back Left | 218 | 0.177 % |
| 2 | African Elephant | Back Right | 225 | 0.183 % |
| 3 | African Elephant | Front | 5,861 | 4.771 % |
| 4 | African Elephant | Front Left | 3,643 | 2.965 % |
| 5 | African Elephant | Front Right | 3,855 | 3.138 % |
| 6 | African Elephant | Left | 1,073 | 0.873 % |
| 7 | African Elephant | Right | 1,031 | 0.839 % |
| 8 | Giraffe | Back | 197 | 0.160 % |
| 9 | Giraffe | Back Left | 1,549 | 1.261 % |
| 10 | Giraffe | Back Right | 1,491 | 1.214 % |
| 11 | Giraffe | Front | 2,708 | 2.204 % |
| 12 | Giraffe | Front Left | 1,899 | 1.546 % |
| 13 | Giraffe | Front Right | 1,934 | 1.574 % |
| 14 | Giraffe | Left | 12,344 | 10.048 % |
| 15 | Giraffe | Right | 12,427 | 10.115 % |
| 16 | Negative | - | 18,018 | 14.666 % |
| 17 | Grevy's Zebra | Back | 221 | 0.180 % |
| 18 | Grevy's Zebra | Back Left | 1,513 | 1.232 % |
| 19 | Grevy's Zebra | Back Right | 1,523 | 1.240 % |
| 20 | Grevy's Zebra | Front | 344 | 0.280 % |
| 21 | Grevy's Zebra | Front Left | 342 | 0.278 % |
| 22 | Grevy's Zebra | Front Right | 365 | 0.297 % |
| 23 | Grevy's Zebra | Left | 9,305 | 7.574 % |
| 24 | Grevy's Zebra | Right | 9,365 | 7.623 % |
| 25 | Plains Zebra | Back | 199 | 0.162 % |
| 26 | Plains Zebra | Back Left | 755 | 0.615 % |
| 27 | Plains Zebra | Back Right | 691 | 0.562 % |
| 28 | Plains Zebra | Front | 40 | 0.033 % |
| 29 | Plains Zebra | Front Left | 357 | 0.291 % |
| 30 | Plains Zebra | Front Right | 385 | 0.313 % |
| 31 | Plains Zebra | Left | 14,231 | 11.584 % |
| 32 | Plains Zebra | Right | 14,577 | 11.865 % |
| | | **Total Patches** | **122,854** | |
| | | **Training Patches** | **92,140** | |
| | | **Test Patches** | **30,714** | |

Table 7: patch dataset category (species and viewpoint) distribution



**Linear** **H1**



**H2** **H4**

Figure 4: confusion matrices for the patches network architectures