Deep Learning for Image Classification

Ashley Jacobson
apjacobson@calpoly.edu

Markelle Kelly
mkelly23@calpoly.edu

Jenna Landy
jlandy@calpoly.edu

Abstract: Transfer learning, which reuses part of a pre-trained neural network on a new problem, is one method for mitigating the long training times of complex networks and is useful for smaller datasets without enough observations to fully train their own network. We explore this concept on an image classification problem, training a network on the CalTech 101 categories dataset [1] and then using all but its last layer to distinguish between photos as taken by Alex Dekhtyar or Hunter Glanz. As a comparison, we also perform transfer learning using VGG16, a much deeper and more accurate network developed at the University of Oxford [2].

Introduction

Training convolutional neural networks, while a powerful strategy for image classification, often puts a strain on valuable time and resources. Further, these complex models require a lot of data to train sufficiently and there are not always enough observations to do so. However, once trained, the initial layers of a network can be useful for a variety of classification problems. When a network attempts to classify an image, it first must extract some features from the image, then make sense of these for the actual classification at hand. Transfer learning takes advantage of this: train a network to classify one set of images, then reuse it on other image sets, simply replacing the dense layers at the end. In this way, we only have to train the final layers that perform the actual classification, skipping the much more intensive convolutional feature extraction.

To explore this, we trained a convolutional neural network to classify images from the CalTech 101 dataset [1], which includes 101 categories, such as "rhino," "chair," and "helicopter." We then used transfer learning to apply this network to the AlexHunter dataset of photographs taken by Dr. Dekhtyar and Dr. Glanz. To evaluate our results, we created another model with transfer learning, this time using the VGG16 network, which is much more complex and was trained on the 1000-category ImageNet dataset [2]. Based on this, we examined our model in terms of size, training time, and predictive accuracy.

Dataset Creation

Our `DataPreparation.ipynb` notebook defines a function to load the data on command. We initially planned to save the dataset to a CSV, but found that opening a file of this size took longer than reloading the entire dataset from scratch. Further, this function takes height and width as parameters, so it was easy to test out different image sizes to see how this affects training time and accuracy. This function opens all photos from the CalTech 101 categories and AlexHunter datasets, saving the resized image arrays and the category names. Figure 1 shows an example of one data point as an image array for `X` and a category name for `y`.

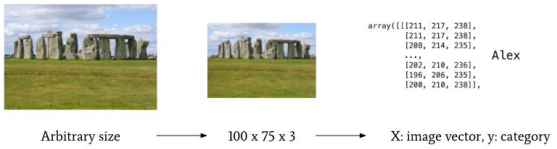

Figure 1: Image Processing

We decided to resize all of the images to 100 x 75 x 3. This kept the original proportion of most images in the AlexHunter dataset and shrunk them to a point that our models would run in a reasonable amount of time. We ensured that the images were still recognizable to the human eye by visually inspecting examples from each category.

CNN for CalTech101

Implementation

In building a convolutional neural network for the CalTech 101 categories dataset, we trained on a random 70% of the data and tested on the other 30%. We first used trial and error to decide on the basic structure displayed in Figure 2.

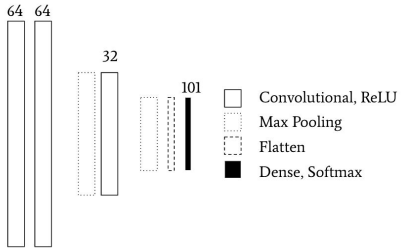

Figure 2: Basic CNN Architecture

This network structure starts with two convolutional layers with 64 filters each using the ReLU activation function, followed by a max pool layer, another convolutional layer of 32 filters, another max pool layer, then a softmax flattening layer. The final dense layer with 101 nodes is the only layer specific to the CalTech dataset. We chose this relatively simple structure instead of a deeper network to allow us to better explore optimization.

Optimization

This network is small enough that we were then able to optimize batch size, learning rate, and momentum with a grid search. We performed a grid search to test batch sizes of 50 or 100, learning rates between 0.00005 and 0.0003, and momentums between 0.01 and 0.05. Due to time constraints, we trained each model for only 14 epochs. All together, even with a limited number of epochs and a relatively simple model, the search took over 36 hours to run.

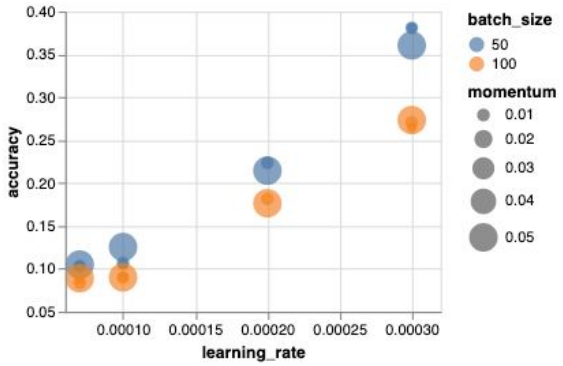

Figure 3: Grid Search Optimization

Figure 3 demonstrates that, overall, higher learning rates improve 14-epoch accuracy. Smaller batch sizes are also associated with higher accuracy. Momentum does not seem to have a dramatic impact in general, but a smaller momentum did improve the network with the higher learning rate and smaller batch size.

Evaluation

Of all the combinations tested, the best network used a batch size of 50, a learning rate of 0.0003, and a momentum of 0.006. This final CNN had a test accuracy of 38.12% on the CalTech 101 categories dataset.

Transfer Learning using CalTech101

Implementation

To perform transfer learning from our trained CNN to the AlexHunter dataset, we first copied the model and removed the last dense layer. We froze these layers by setting their `trainable` parameter to `False`. This means that when we train the new classifier, the weights for our

pretrained layers remain constant. We then added a final dense layer with two nodes—one for Hunter and one for Alex. Again, we trained on a random 70% of the data and tested on the remaining 30%.

Evaluation
Training this model with stochastic gradient descent for three epochs with a batch size of five, learning rate of 0.01, and no momentum resulted in a test accuracy of 99.1%, with one incorrect prediction.

Transfer Learning using VGG16
Implementation
We decided to compare transfer learning using our trained CNN to that using a state-of-the art image classification model to see if ours performed similarly in distinguishing the photographer in the AlexHunter dataset.

The VGG16 model was proposed in the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman. It is an extremely accurate image classifier, with an accuracy score of 92.7% on the ImageNet dataset of over 14 million images in 1000 classes. This pretrained model can be accessed with Keras in the applications module.

We performed the same transfer learning procedure as we did with our neural net, training the last dense layer on a random 70% of the data and testing on the remaining 30%.

Evaluation
Training this model with stochastic gradient descent for three epochs with a batch size of ten, learning rate of 0.01, and no momentum resulted in a test accuracy of 99.1%, with one incorrect prediction.

Results
We compared classification results on the AlexHunter dataset using transfer learning both from the CNN we built on the CalTech 101 categories dataset and the pretrained VGG16 model in terms of accuracy and complexity.

Accuracy
Using our network in transfer learning resulted in a final AlexHunter test accuracy of 99.1%, with one image misclassified. The confusion matrix in Figure 4 shows that this misclassification was a picture taken by Alex that was predicted to be taken by Hunter (image shown in Figure 5). This is likely due to the street in the foreground as well as the tree leaves on the side, both of which are common in Hunter's images. However, we were a bit surprised because there is a person in the image—an unusual feature for the Hunter category.

		true	
		Alex	**Hunter**
predicted	**Alex**	54	0
	Hunter	1	55

Figure 4: Confusion Matrix for AlexHunter using our CNN

Figure 5: Misclassification for AlexHunter using our CNN

VGG16's accuracy on the AlexHunter dataset also was 99.1%. The confusion matrix in Figure 6 shows that this network also misclassified only one image, but this time predicting one of Hunter's photos as Alex (image shown in figure 7). We suspect this image was difficult to classify due to the presence of a person, as well as the road being half in the shadows and half in the sun.

		true	
		Alex	**Hunter**
predicted	**Alex**	55	1
	Hunter	0	54

Figure 6: Confusion Matrix for AlexHunter using VGG16

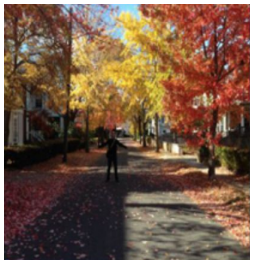

Figure 7: Misclassification for AlexHunter using VGG16

Complexity

In terms of training time and size, our model was by far superior to VGG16. While for our network, each epoch ran in about 13 seconds, those of VGG16 took several minutes. The network pulling from VGG16 was also 537 megabytes—over 1,000 times bigger than our network (444 kilobytes).

Given that our model is much more lightweight than VGG16, we are very pleased that it attained the same accuracy.

Figure 8 summarizes the results of all three models built.

	Dataset	Transferred From	Test Accuracy
Model 1	CalTech 101	N/A	0.3812
Model 2	AlexHunter	Model 1	0.991
Model 3	AlexHunter	VGG16	0.991

Figure 8: Model Summaries

Conclusion

Overall, we found that it is not that difficult to distinguish between Hunter and Alex photos. This is likely because of consistent patterns in Hunter photos: once the network picks up on a few key features, it can recognize these pictures very reliably.

One big takeaway from this project is that, when it comes to simple image classification problems, a deeper neural network is not always better. Even though our network only attained 38% accuracy on the CalTech dataset, it was perfectly sufficient for AlexHunter classification, reaching the same test accuracy as the state-of-the-art VGG16 classifier.

If we were to try a more difficult classification problem, we would want additional computing power. Being able to train models more quickly would enable us to run them for more epochs and explore more complex networks. It would also be interesting to train our CNN on a different dataset, especially one that includes more landscapes. The images in the CalTech dataset mainly pictured individual objects; we might be able to achieve better accuracies if a larger variety of image types were represented.

Sources

[1] L. Fei-Fei, R. Fergus and P. Perona. *Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories*. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. 2004

[2] Hassan, Muneeb ul. "VGG16 - Convolutional Network for Classification and Detection". *Neurohive*. https://neurohive.io/en/popular-networks/vgg16/