Semantic Representations for Image Classification

Davis Liang
A10648728
d1liang@ucsd.edu

Arvind Rao
A10735113
a3rao@ucsd.edu

Daniel Riley
A10730856
dgriley@ucsd.edu

Cuong Luong
A10642826
c3luong@ucsd.edu

Gannon Gesiriech
A11479234
ggesirie@ucsd.edu

Abstract

Many commonly-used datasets for image classification benchmarking, such as Caltech and CIFAR, contain classes in which a vast majority are not by nature orthogonal. In this work, we introduce the Semantic ConvNet (SCNN) which exploits the inherent relationships between such classes by regressing on word2vec outputs on top of a traditional convolutional neural network. We show how the Semantic ConvNet is as accurate and robust to error as a classical CNN using the CIFAR-10, CIFAR-100, Caltech-101, and Caltech-256 datasets. In addition, we demonstrate the Semantic ConvNet's advantage over the classical CNN from its ability to generalize to new classes.

Introduction

Word embedding models transform words into multidimensional vectors capable of capturing their semantic meaning. These word embedding vectors capture the relationships between words through their locations in a higher dimensional vector space. Effectively, words that are closer in semantic meaning or used in similar contexts are mapped to spatially similar points in the word embedding space.

Classification of images is typically done through deep convolutional neural networks trained through backpropagation in which the target for each image is a one-hot encoding of the true class. The corresponding objective function to optimize is categorical cross entropy. Setting class labels as one-hot encodings relies on the assumption that all of the classes are orthogonal. However, the classes within a vast majority of image datasets are not by nature orthogonal. For example, the classes within Caltech-256 are semantically related with each other (this similarity between classes is shown in the visualization section) and we plan on exploiting these inherent relationships by modifying the targets to be word embedding vectors drawn from the word2vec model. The objective is to explore the effectiveness of training image classifiers through regression on word2vec word embedding targets.

We use a pretrained VGG-16 network (Simonyan, Zisserman 2015) as the base model for our Semantic CNN. In addition, we use Google's word2vec model (Mikolov 2013) to acquire the word embedding vectors for each class label. We compare our results to those from a classic CNNs trained on CIFAR-10 (Krizhevsky 2009), CIFAR-100 (Krizhevsky 2009), Caltech-101 (Griffin 2007), and Caltech-256 (Griffin 2007). We have chosen to do a relative comparison (where we train the Semantic and classic CNN side-by-side with the same hyperparameters) due to limitations of both time and hardware. Given more time and computational power, we could apply hyper-parameter searching techniques to tune our network and compare the performance or ensemble with state-of-the-art models.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Background

Many state-of-the-art convolutional neural network models (Krizhevsky 2012, Zeiler 2013; Sermanet 2014, Simonyan 2014) do not comment about their incentive to use one-hot targets. For some early datasets like MNIST (Lecun 1998) class categories are naturally orthogonal. For other datasets such as CIFAR-10 (Krizhevsky 2009), CIFAR-100 (Krizhevsky 2009), Caltech-256 (Griffin 2007), and Caltech-101 (Griffin 2007), one-hot encoding vectors are convenient approximations and are simple to work with. We found two types of semantic encoding to be robust, flexible, and convenient enough to work with: word2vec (Milokov, Chen, Corrado, Dean 2013) and GloVe (Pennington et. al 2014). While both of these were usable, due to availability we chose to use word2vec. The most popular pre-trained word2vec variant is trained on Google News Data (Milokov, Sutskever, Chen, Corrado, Dean 2013). This is the semantic represenation model that we will use for the rest of this paper. By combining the power of classic convolutional neural networks and the additional information provided by the word2vec model, we forsee several benefits. We cover these benefits in detail in our Experiments section. Finally, due to the slightly non-conformist nature of our model, the Semantic CNN, prior work in the same vein was hard to come by.

Due to constraints with time and computational power, we built our model on top of a pre-trained VGG-16 variant trained on ImageNet (Russakovsky 2014). In the same vein of logic, we removed Imagenet from the list of datasets that we train on. Our only modification to VGG-16 is on the final prediction layer.

Word2Vec Visualization

To get a better understanding of how valuable the word2vec model really is, we have created several visualizations. First, we have conducted principal component analysis (PCA) on 10 word embedding vectors from 10 classes within CIFAR-10 and Caltech256 and plotted their vectors relative to each other. Next we took images from two of the the datasets CIFAR-10 and Caltech256 and found the cosine similarity and normalized cross correlation between images.

PCA is a vector dimensionality reduction technique with the following steps: (1) center the data by subtracting the mean, (2) calculate the covariance matrix, (3) calculate the n largest eigenvalues and eigenvectors of the covariance matrix, (4) normalize the eigenvectors corresponding to the n largest eigenvalues

In our first visualization technique we conduct PCA on 10 vectors from CIFAR-10 and Caltech-256 with dimensionality reduction to $n = 2$ dimensions (Figure 1). We see that this provides a good intuition into how related the vectors are. In CIFAR-10 we see that cat, dog and horse have a large cosine similarity whereas frog and deer have a much smaller cosine similarity. This comes back to the contextual nature of word2vec. The large cosine similarity between cat, dog, and horse is

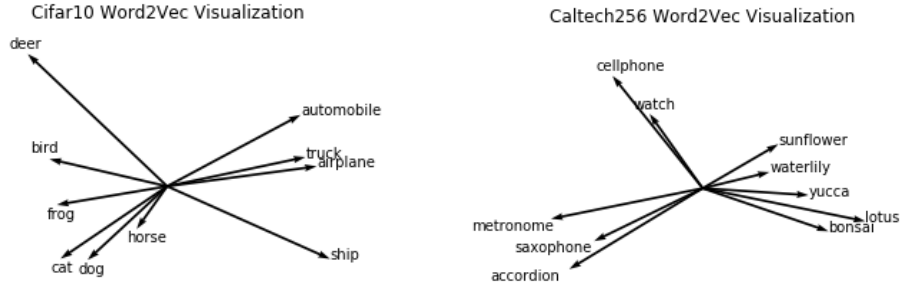


Figure 1: Class label word2vec visualizations

Figure 2: CIFAR-10 im2vec Visualization with Normalized Cross Correlation and Cosine Similarity (The bottom four images correlated against top image)

probably attributed to them being quadrupeds, mammals and common pets. Despite the deer being a mammal and quadruped it is much less known to be domesticated. The cosine similarity of the frog and the cat, dog and horse (CDH) is larger than that of the deer and CDH. This is possibly attributed to the frog being a quadruped and a pet but not a mammal.

As a precursor to the second technique it is a good idea to discuss what the normalized cross correlation and the cosine similarity are and why we were motivated to compare. Normalized cross correlation is a two step technique using the pixel values of the two images. First we normalize the images so that the lighting conditions for each do not effect the correlation. This involves first subtracting the mean of each image then dividing by the standard deviation. Cosine similarity is a technique designed to find the cos angle between two vectors. For our situation we use the cosine similarities between class label word2vec outputs.

We are interested in the correlation and cosine similarity to see how visually and semantically similar these images/classes are. In the CIFAR-10 dataset we see that the largest image correlation is between the plane and truck and the smallest is between the frog and truck.

We suspect that a possible contributor to the high correlation is the similarity in relative size, orientation (direction facing), and attributes of the plane and truck. On the other hand, we see that the frog has the opposite orientation and is much wider leading to a small correlation.

If we analyze the cosine similarity between the vectors associated with each class we see that between the images we selected, the highest similarity is between the truck and plane, with truck and car being a close second. This is very interesting since we expected the vectors for the car and truck to be extremely close since they both have four wheels and are used for transportation. On the other hand we have a plane and truck. They are much more different aesthetically in comparison between the truck and car but closer contextually. All of these vehicles are used to transport people but a connection the car does not hold is that both a truck and a plane are commonly used to transport goods other than people. This shines a light on the robustness of each of the vectors in the word2vec dictionary.

Correlation = .33 Cos Similarity = .18	Correlation = .45 Cos Similarity = .13	Correlation = .6 Cos Similarity = .1	Correlation = .19 Cos Similarity = .23

Figure 3: Caltech256 im2vec Visualization with Normalized Cross Correlation and Cosine Similarity (The bottom four images correlated against top image)

Model

Architecture

Our model, shown in figure 4, is kept extremely simple for ease of both implementation and understanding. Our base model is VGG-16 pretrained with ImageNet weights. We remove the top softmax layer from the pretrained model and replace it with a custom regression layer. The difference between the classic CNN and the Semantic CNN are different output unit activation functions and counts. The output units of the Semantic CNN are linear and has 300 dimensions. The prediction can be extracted from this 300 dimensional output vector by choosing the word vector in our dictionary of targets with smallest cosine distance. The output units of the classic CNN are softmax units and of N-dimensions, where N is the number of categories in the dataset used. The inputs to each model will be images of size 224x224x3. Thus, for different datasets, we must resize the images to meet these input shape requirements. We train each model to a specified number of epochs as convergence will take too long.

The units for the Semantic CNN are linear because the semantic embeddings from word2vec can be negative (thus we cannot use softmax). We chose to run experiments using (1) PreLU (2) tanh and (3) linear units and found that linear units provided the best results empirically.

For the Semantic CNN, we replaced one-hot encodings of the labels with 300 dimensional word embeddings by querying from the word2vec model trained on Google News. Each of these embeddings were generated by querying from the metadata file in the official CIFAR-10 and CIFAR-100 directories.

Training

The loss for the Semantic CNN is cosine proximity.

$$\text{Cosine proximity}(x, y) = 1 - \frac{x \cdot y}{\|x\|\|y\|}$$

while the loss for the classic CNN is categorical cross entropy. We optimize with the Nadam optimizer which is just Adam with Nesterov momentum. We implement everything on Keras with a Tensorflow backend in Python 2. All software we used is both extremely well documented and extremely popular (one can find documentation by a simple Google search). Additionally, the

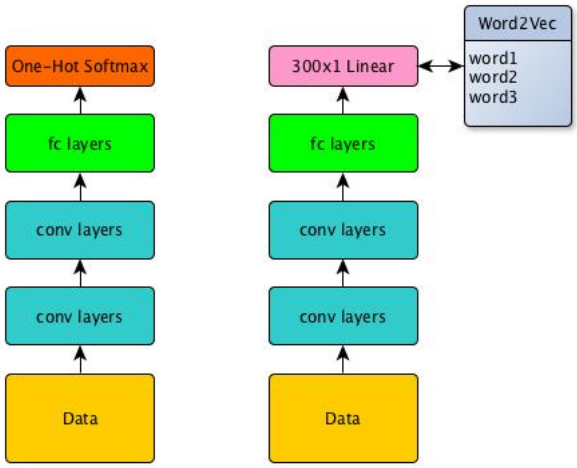


Figure 4: Classic CNN (left) vs. Semantic CNN (right)

software can be obtained using a simple `pip install`. Lastly, our hardware consists of an Nvidia K80 GPU, 64GB of RAM, and a sufficiently powerful Xeon CPU.

Datasets

We use the CIFAR-10, CIFAR-100, Caltech-101, and Caltech-256 datasets for our experiments. CIFAR-10 contains 10 categories of 60000 32x32 images. CIFAR-100 contains 100 categories of 60000 32x32 images. Caltech-101 contains 101 categories of about 40 to 800 images in each category of varying sizes. Caltech-256 is similar to Caltech-101 but with 256 categories. For the datasets, we train on 80% of the images and validate on 20%.

Cleaning was necessary for a small number for the categories within the Caltech and CIFAR datasets. This involved cross-checking the categories used in the datasets with the pretrained word2vec model. We acknowledge that some categories may not be represented in word2vec. We can either select a synonymous word representation or merge categories for these categories, as detailed in the next paragraph.

Categories such as 'Flamingo' and 'Flamingo-Face' and 'Face' and 'Face-hard' in Caltech were combined into single categories of 'Flamingo' and 'Face', respectively. Another common cleanup method was to modify the category name to a similar word that existed in our pretrained word2vec model. As a fabricated example, 'BMW-I7' would be converted to 'car'. Specifically for CIFAR-100, 'Aquarium Fish' was renamed to 'Fish', and 'Sweet Pepper' to 'Sweet Peppers'. This is again to make sure that all categories being considered have a word2vec representation. As a result of this preprocessing, Caltech-256 was left with 250 categories, Caltech-101 was left with 95 categories, and CIFAR-10 and CIFAR-100 still had 10 and 100 categories respectively.

Experiments and Results

We construct a set of experiments to test three specific metrics of network performance. We argue that a network should be (1) accurate, and (2) robust to error, and (3) generalizable to new data. We argue that our model, Semantic CNN, satisfies both (1) and (2) and in some situations surpasses a classic CNN implementation in its ability to generalize to new data (3). These two models are comparable given the heuristics we chose for predicting targets (see the training section). To be specific, the Vanilla CNN model's predictions are extracted directly from the softmax layer. The Semantic CNN's predictions are extracted from this 300 dimensional output vector by choosing the

word vector in our dictionary of targets with smallest cosine distance.

(1) Accuracy

In the first set of experiments, we show that our Semantic CNN has similar classification performance compared to a classic CNN. We do this by fine-tuning VGG-16 (by removing the top layer and appending a new top layer) on 4 different datasets for both semantic and one-hot labels. This amounts to 8 experiments in total. For the semantic regression model, we use a linear output activation because word2vec can have both negative and positive values. We found that a linear activation has better empirical performance than both PreLu and tanh. Again, we choose not to use softmax because our task is regression on a vector that can potentially have negative elements. For the one-hot labels we used the traditional softmax outputs. However, our model performs regression and outputs a vector in the word embedding space so we have to perform nearest neighbors across all class label word embedding vectors to predict a category for an input. Each experiment consisted of 15 epochs of training with the same hyperparameters (Batch size of 200, default Nadam optimizer parameters, LeCun uniform initializations, and no data augmentation/regularization).

Our network shown to be just as powerful as a classic CNN. The results from this first experiment show that the Semantic CNN and the classic CNN arrives at similar final converged accuracies in every single dataset as shown in the first column of figure 6. From our results on 4 different image datasets, we show that the Semantic CNN either beats or performs very closely to a classic CNN.

(2) Robustness

In the second experiment, we show that the Semantic CNN is just as robust as classic CNNs. We define 'robustness to error' by showing, for the datapoints that the CNN predicted wrong, that the correct answer existed in the top-K (this means that even though the semantic CNN was wrong, it had still learned reasonable features for that category. Additionally, the error may have been reasonable, i.e. predicting grass instead of park). For the Semantic CNN we choose top-K by taking the predicted regression vector and choosing the K closest vectors in the categories dictionary. For the classic CNN, we choose the top-K by taking the top-K softmax activations. We measure robustness by showing that the accuracy will go up just as much for the Semantic CNN as for the classic CNN if we take the top-N categories. This is shown in figure 6

(3) Generalizability

In the final experiment, we show that the Semantic CNN is more generalizable to new data than a classic CNN. Obviously, to include a new class in a classic CNN, we must at the very least add a node to the output softmax layer and retrain the net. We show that we can include a new class in the Semantic CNN and be able to get reasonable results without retraining or augmenting our old pre-trained network in any way.

Images from new classes were fed into the network and the resulting output vector was classified according to the top 5 nearest classes according to cosine similarity. We tested this generalization approach by adding classes from Caltech-256 to Caltech-101. The results revealed a high generalization rate with over half of the added classes achieving an accuracy greater than random guessing. The performance of each class varied according to the relative distance of the two classes

New class	screwdriver	harpsichord	lawn mower	teddy bear
Accuracy	59%	71%	57%	0%

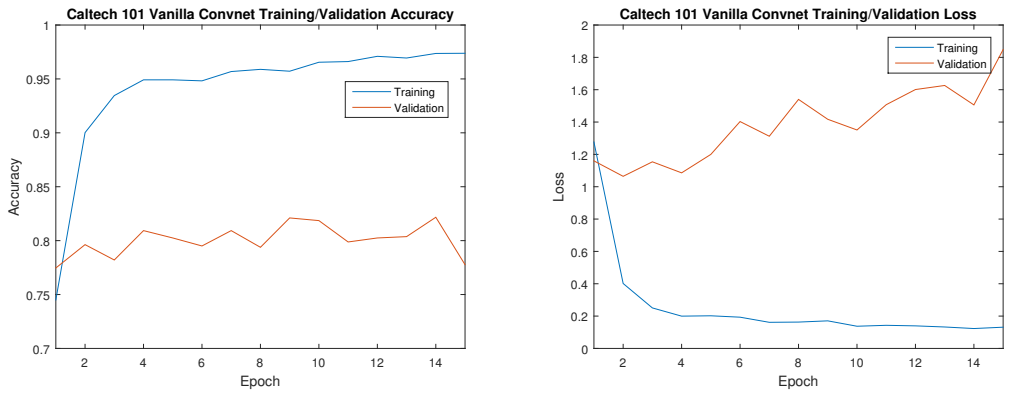
Figure 5: Caltech-101 Generalization examples (Semantic CNN top-5 accuracy on completely new class without further training.)

ConvNet	Top-1	Top-2	Top-3	Top-4	Top-5
Semantic ConvNet Caltech-101	0.82	0.89	0.91	0.92	0.93
Vanilla ConvNet Caltech-101	0.78	0.87	0.91	0.92	0.94
Semantic ConvNet Caltech-256	0.67	0.75	0.79	0.81	0.83
Vanilla ConvNet Caltech-256	0.66	0.76	0.80	0.82	0.84
Semantic ConvNet CIFAR-10	0.74	0.87	0.93	0.96	0.98
Vanilla ConvNet CIFAR-10	0.7	0.86	0.93	0.96	0.98
Semantic ConvNet CIFAR-100	0.49	0.60	0.66	0.70	0.73
Vanilla ConvNet CIFAR-100	0.42	0.55	0.62	0.67	0.70

Figure 6: Top-K accuracies from final models

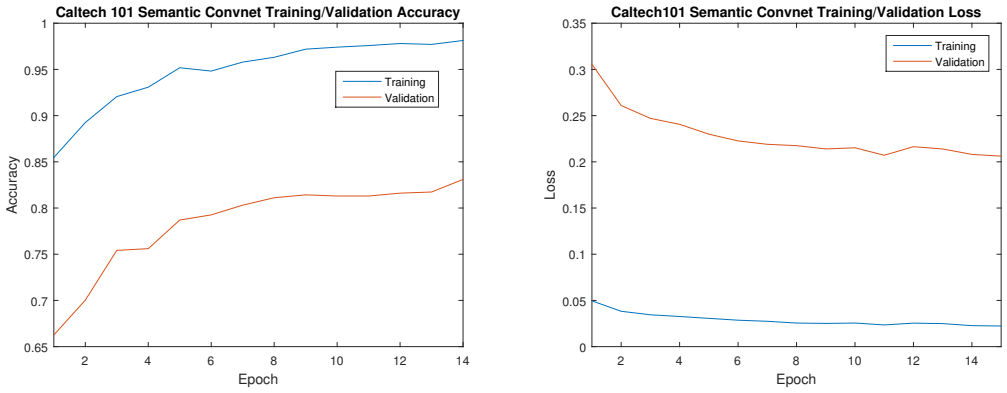
and the two images. The best accuracies were taken from classes that were close to classes on which the network trained.

Examples of this are screwdriver, harpsichord, and lawn mower, which give top-5 accuracy values of 59%, 71% and 57%. However, one example of poor generalization was the class 'teddy bear' which received an accuracy of 0. 63% of the time the network's top prediction was 'panda'. According to word2vec, the similarity between these two words is about 0.34 which is quite low. This is because, while they look visually very similar, the way that panda and teddy bear are used semantically differs greatly. Because of this, the network classifies the teddy bear image to be near the panda vector and away from the actual label.



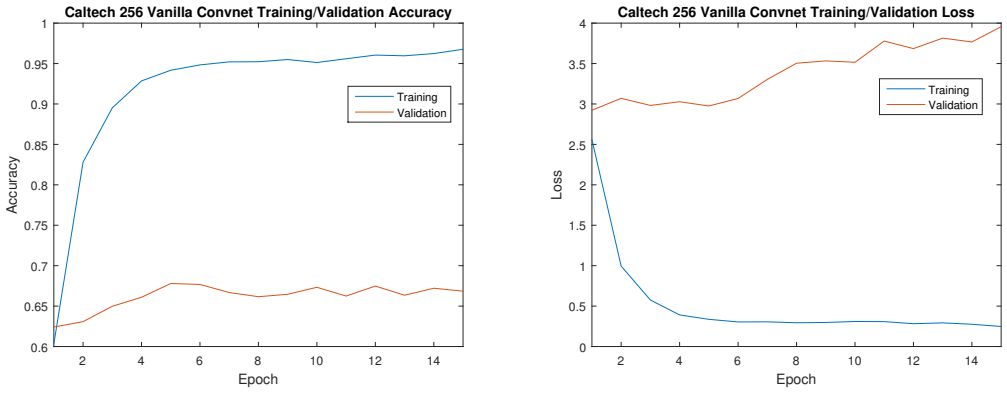
(a) (b)

Figure 7: Caltech 101 Vanilla ConvNet Training/Validation Loss/Accuracy



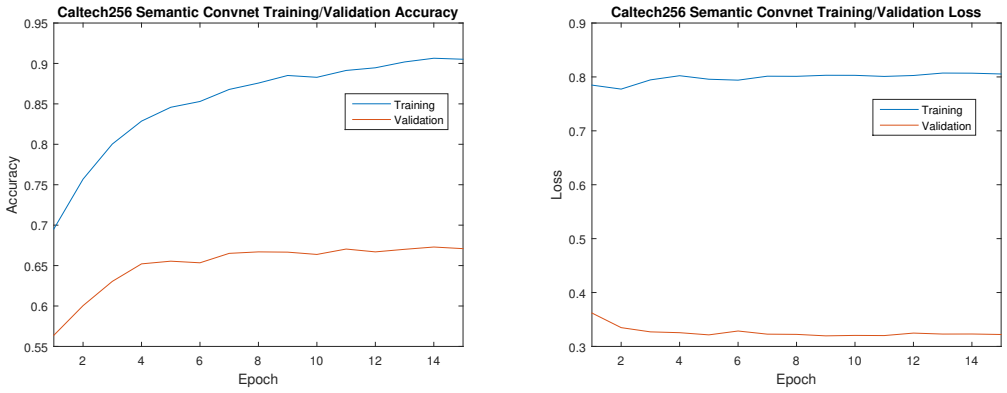
(a) (b)

Figure 8: Caltech 101 Semantic ConvNet Training/Validation Loss/Accuracy



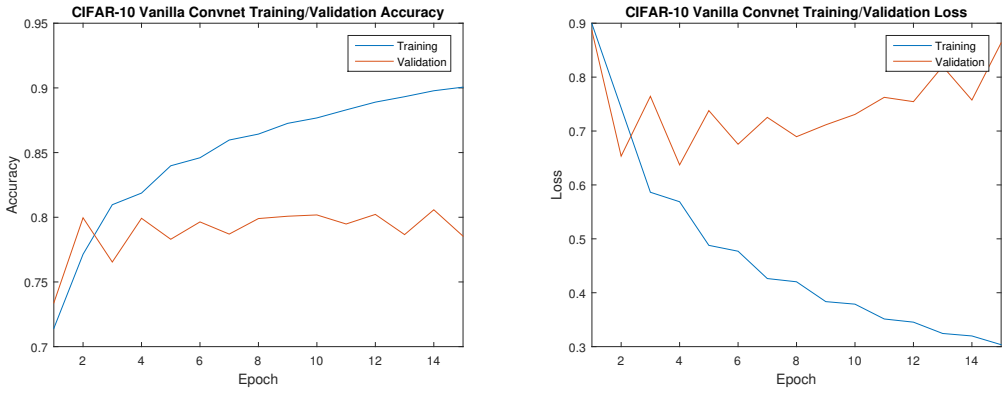
(a) (b)

Figure 9: Caltech 256 Vanilla ConvNet Training/Validation Loss/Accuracy



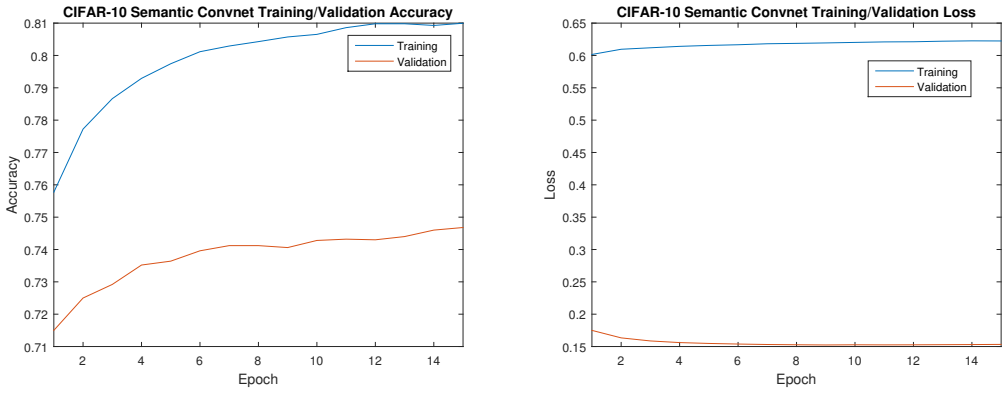
(a) (b)

Figure 10: Caltech 256 Semantic ConvNet Training/Validation Loss/Accuracy



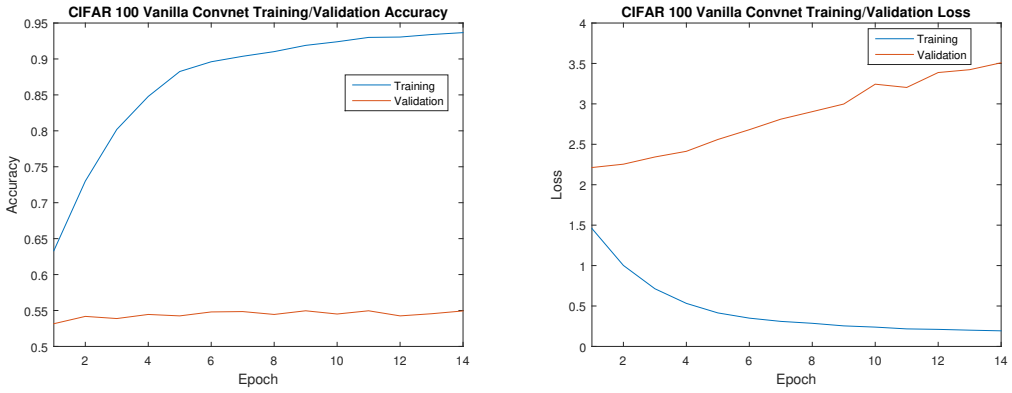
(a) (b)

Figure 11: CIFAR-10 Vanilla ConvNet Training/Validation Loss/Accuracy



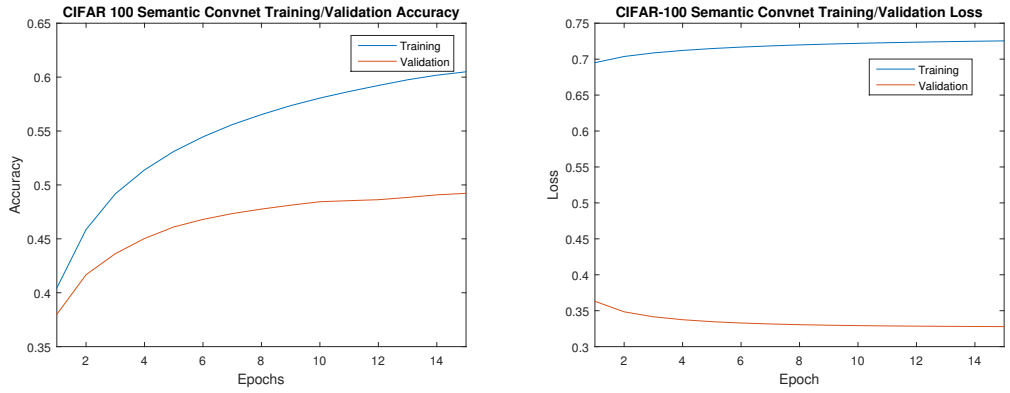
(a) (b)

Figure 12: CIFAR-10 Semantic ConvNet Training/Validation Loss/Accuracy



(a) (b)

Figure 13: CIFAR-100 Vanilla ConvNet Training/Validation Loss/Accuracy



(a) (b)

Figure 14: CIFAR-100 Semantic ConvNet Training/Validation Loss/Accuracy

To easily compare classification performance we have generated Confusion Matrices for both one-hot and semantic on our model trained with the CIFAR-10 dataset. If we compare the main diagonals of the one-hot and the semantic it quickly stands out to us that semantic predicts the right value more for every class except horse. We also see that for one-hot, automobile is predicted as truck more often than automobile. This reinforces our claim that vectorization can lead to a more accurate prediction despite having visually similar data due to its innate ability to contain more information in comparison to the one-hot alternative.

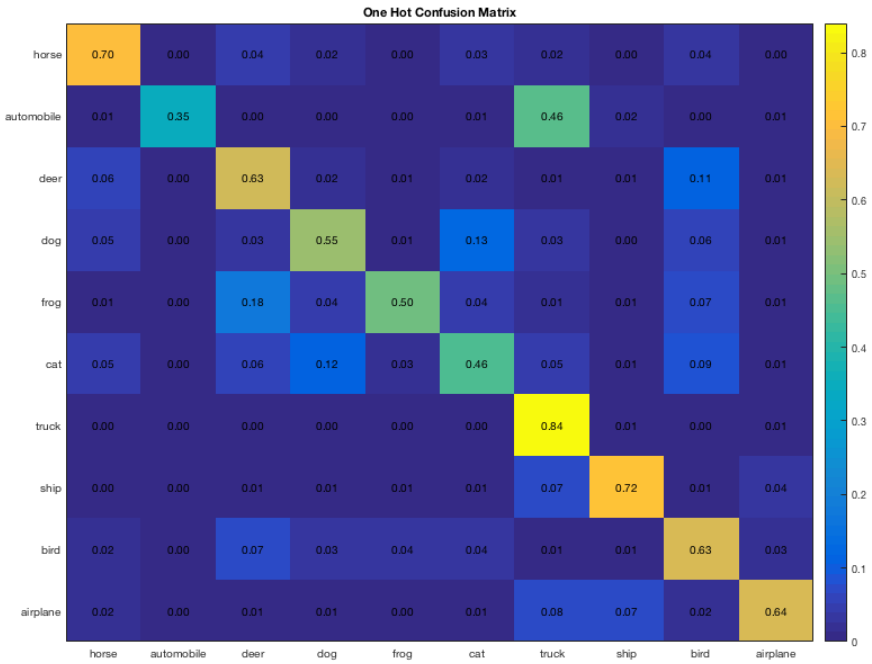


Figure 15: CIFAR-10 One-Hot Confusion Matrix

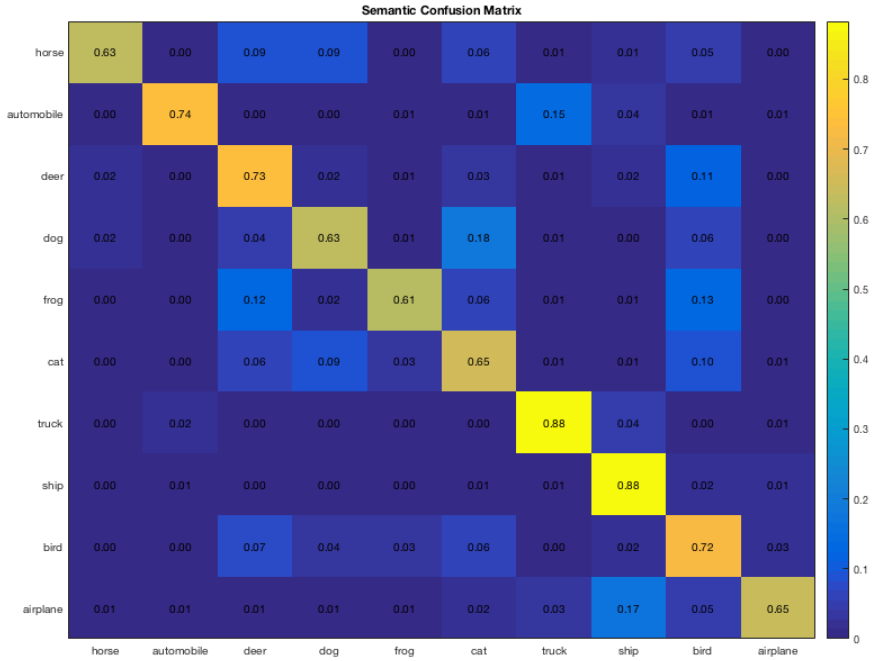


Figure 16: CIFAR-10 Semantic Confusion Matrix

Discussions and Conclusions

Analysis of the experimental results of Semantic CNN brings up a couple of interesting points. To begin, we were genuinely surprised by the accuracy of our semantic models. In many cases, the Semantic CNN beat the classic CNN in image recognition tasks. After performing our experiments and corresponding data visualization tasks, we came to a further conclusion about the definitive power of word vector labels. First, word vector labels provide more information than one-hot encoding vectors. Second, as the number of categories increase, the ability of the network to use word vector labels also increases (i.e. the network establishes a larger vocabulary and can use this vocabulary to generalize to new unseen labels). Third, in some cases word vector labels trained on natural language tasks may not be optimal for vision tasks (for example, the vector for deer and horse are far apart even though they look fairly similar. This difference is a result of the different contexts in which horses and deer show up in text).

With this in mind, we reach the following conclusions that support our prior hypotheses. First, our model clearly never underperforms relative to a classic CNN. Second, our model clearly has stronger generalizability than a classic CNN. Finally, our model is just as robust to error relative to a classic CNN.

Concept Novelty

Our Semantic CNN is a completely new architecture inspired by convolutional neural networks and word2vec. We applied the architecture to 4 well-known datasets such that further experimentation could be compared to state-of-the-art results. Additionally, these datasets were popular enough that most variations are well-documented and organized. Our primary innovation consists of finding a better way to construct the output target vectors which would supply additional semantic meaning to target outputs rather than just orthogonal vectors. We provide a comprehensive suite of analysis of

our results via both visualization of our semantic vectors as well as the design of our experiments. Our model was novel and we were hard-pressed in finding similar work.

Code

The code that we have worked is available at `https://github.com/xsaardo/Semantic-Convnet`

Contributions

- Cuong worked on creating the semantic and vanilla CNN models in Keras and recording metrics.
- Davis processed the word2vec model and did some exploratory experiments on CIFAR-100 on VGG16. Additionally, I proposed the original idea of the Semantic CNN and came up with the experiments and visualization (confusion matrix, PCA on word2vec vectors, etc.).
- Arvind worked on training and reporting the semantic and vanilla CNN models accuracies.
- Daniel preprocessed Caltech-256, created word2vec visualizations, and worked on adapting semantic CNN models to new datasets.
- Gannon worked on the preprocessing of CIFAR 10 and word2vec visualization.

References

A. Krizhevsky, 2009, Learning multiple layers of features from tiny images, Master's Thesis, Department of Computer Science, University of Toronto

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

Griffin, G., Holub, A., Perona, P. (2007). Caltech-256 object category dataset (Technical Report 7694). California Institute of Technology. http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014): n. pag. Web.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A.,Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. CoRR, abs/1409.0575, 2014.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR, abs/1312.6229.

Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556.

Zeiler, Matthew D., and Rob Fergus. "Visualizing and Understanding Convolutional Networks." Computer Vision ECCV 2014 Lecture Notes in Computer Science (2014): 818-33. Web.