ABSTRACT

This paper is concerned with creating deep neural networks and to analyze how various factors affect the fitting and ultimate test set performance of these networks. Besides building a deep neural network with more than one hidden layer, alternative network structures/topologies – namely Convolutional Neural Networks (CNN), are explored. The key assignment objective is for us to gain hands-on, practical experience with understanding the neural network transition from simple (i.e., single hidden layer) to deep (i.e., multiple hidden layers) networks. Drawing from the learning of this exercise, management recommendation is provided for developing a CNN model for facial recognition of mobile phone users.

1. INTRODUCTION

In this exercise various neural network topologies derived from Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN) are built, fitted, and tested in the Python coding framework. Attempts were made to compare and evaluate alternative network structures/topologies, such as 1) dense versus convolutional neural networks, 2) convolutional neural networks with differing structures in terms of convolution and pooling layers, and 3) neural networks with or without regularization. CNN, in particular, have been proven very effective in areas such as computer image recognition and classification. While DNN operation was discussed in the previous assignment, CNN are newly introduced through this assignment. The CNN architecture utilized in this assignment illustrates four main operations in figure 1: convolution, non-linearity (ReLU), pooling or sub sampling, and classification (fully connected layer). These operations are the fundamental building blocks of CNN, and they warrant a detailed discussion in the Methods section.

The DNN and CNN architecture/hyperparameter variations are examined in this paper for each of their training requirement and performance in classifying CIFAR-10 data (Krizhevsky, 2009, p. 32-35) which consists of a set of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Each image is labeled with the class it represents - 10 random images from each class are shown in figure 2.

A total of 10 experiments were conducted: Experiment 1, 2, 5, and 6 are a set of dense neural network models with two or three hidden layers both before and after regularization is applied, whereas Experiment 3, 4, 7, and 8 are convolutional neural networks with two or three convolution layers plus pooling layers both before and after regularization is applied. The remaining Experiment 9 and 10 have one of the regularization hyperparameters varied from the "best" CNN model in Experiment 8. In addition, Experiment 3 further extracts the CNN model outputs from its 2 max pooling layers filters to visualize in a grid as images in order to have "lighted up" regions examined against image features.

In Experiment 8, T-distributed Stochastic Neighbor Embedding (t-SNE) is utilized to reduce the high-dimensional dataset CIFAR-10 dimensionality to allow for visualization of each label class on a 2D graph (van der Maaten and Hinton, 2008).

This assignment allows us to practice building trustworthy deep learning networks by conducting experiments to determine what features are the hidden nodes are learning. Furthermore, the experience gained from building the CNN models extend into a real-world business case of developing a deep neural network model for facial recognition tasks of the user of a mobile device such as a cell phone.

2. LITERATURE REVIEW

Introduction of CNN structure and operation by Walkarn (2016), Challot (2021), Deshpande (2016) are reviewed. The breakthrough of CNN into mainstream image classification was when one model (Krizhevsky et al., 2012) developed by Geoffrey Hinton and colleagues won the ImageNet competition in 2012. Shortcomings of CNN by Liu et al. (2018) from Uber Engineering have also been reviewed.

3. METHODS

Both the DNN and CNN models were built in the Python environment using Keras packages imported from Tensorflow. The DNN models consist of 3072 input nodes reshaped from the (32x32x3) RGB pixels, two or three hidden dense layers with 384 nodes and 10 output nodes corresponding to the 10 image classes. On the other hand, the CNN models consist of multiple convolution/pooling, flattening, and dense layers. Variations of the DNN and CNN models incorporate regularization techniques including Dropout, Batch normalization, L2 Ridge Regression, and Early Stopping. Other Python packages including NumPy, Pandas, and Matplotlib are used to help with data manipulation and visualization, while Scikit learn and additional Keras packages are used to facilitate data pre-processing and model building.

The CIFAR-10 dataset was imported from Keras datasets directly using the load_data function. Both the training set (50,000 images) and the test set (10,000 images) along with their labels were first inspected through exploratory data analysis. The image dimension and label frequency were verified to be as expected. A sample of images in a sample set of “aeroplane, car, and bird” was displayed with corresponding labels in figure 3.

Before the DNN experiments were run, the training set, validation set and test set data were reshaped into an array of shape (45000, 32*32*3), (5000, 32*32*3), and (10000, 32*32*3) respectively. The input data were essentially flattened before feeding into the input nodes. The input data values were normalized between 0 and 1. The DNN models are created by the Sequential class defined in Keras. All hidden layers are "dense" layers, meaning which all the nodes of a layer are connected to all the nodes of the preceding layer. In Experiments 5 and 6 where regularization was applied to the DNN models, batch normalization and dropout layers were included after each dense layer. Moreover, with the presence of early stopping and callback of the model parameters, the number of epochs were increased from 20 to 100 as the risk of model overfitting is covered.

The CNN models comprised of one or more convolutional layers each followed by a pooling layer and then followed by a fully connected layer as in a standard multilayer neural network. The architecture of the CNN is designed to take advantage of the 2D structure of the input image achieved by local connections and tied weights followed by some form of pooling which results in translation invariant features (UFLDL Tutorial, n.d.). The input to the convolutional layer is a m x m (32x32) image where m is the height and width of the image. Each convolutional layer will have increasing number of filters (or kernels) of size n x n (3x3). The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m−n+1. The primary purpose of convolution is to extract features from the input image, as convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. Each feature map is then subsampled with max pooling and subject to ReLU nonlinearity. Pooling reduces dimensionality of each feature map but retains the most important information and serves a few important purposes to the input

representation (Walkarn, 2016). The output from the convolutional and pooling layers represents high-level features of the input image, which are then flattened and connected to the fully connected layer to classify the input image into various classes based on the training dataset. Adding a fully connected layer can possibly allow learning non-linear combinations of these features as well. The regularized variants of the CNN (in Experiments 7-10) have dropout layers added after every max pooling layer; L2 kernel regularizer (ridge regression) added to the fully connected layer; and batch normalization and dropout layers added after the fully connected layer. In the same set up as the DNN counterparts, early stopping and callback of the model parameters were added steps to the model training. The number of epochs were increased from 20 to 100 as the risk of model overfitting is covered. Experiments 9 and 10 are variations of the CNN architecture in Experiment 8: in Experiment 9 the dropout rate was increased to 0.5 whereas in Experiment 10 the dropout rate was decreased to 0.1 to study the effect of this specific hyperparameter to the CNN performance.

The algorithm used to optimize the model weights and biases as per the given data was defined to be stochastic gradient descent. The loss function to be minimized by the optimizer was selected as “categorical cross entropy” commonly used in a multi-class classification problem. The unregularized models were trained through 20 epochs, while the regularized models were set up for 100 epochs with early stopping and callback steps. Across all experiments the training batch size was set to constant at 64. The “evaluate” method was used to evaluate the trained model performance – accuracy and loss - on the test set.

The model training and validation performance was also inspected by plotting the evolution of accuracy and loss in figure 4 as each epoch completes. As with classification problem analysis

in general, a classification report and 10x10 confusion matrix was constructed in a heatmap format (figure 5) for each experiment to visualize the occurrence each predicted value against the true value of the label.

An additional task was performed in Experiment 3 to generate predicted outputs, extract the outputs from 2 filters from the 2 max pooling layers, place them in image grids (figure 6) for visual examination, and analyze how the reduced, convolved features correlate to the original image features in figure 7.

A comparison was summarized between models from every experiment to compare their model attributes/hyperparameters, model performance, and model processing time/requirement.

4. RESULTS

From the EDA and source data description, the training and test label sets contain balanced classes of images across 10 mutually exclusive categories, an indicator of classification accuracy will reflect model success (Chollet, 2017).

The model characteristics, performance (training, validation, test accuracy and loss) of each model across Experiment 1 to 10 are listed in Table 1. It also shows the number of epochs for each model when the training stopped. Comparing the model summary from Table 1 lends insight to understanding how the different DNN and CNN model architecture and CNN hyperparameters affect the performance in image classification.

From Table 1, it is observed that increasing the number of hidden layers for DNN models between Experiment 1 and 2 did not yield any performance improvement, while slightly prolonged the model training time by 2 seconds. In addition, adding regularization to the DNN

models surprisingly reduced their performance as seen from the lower test accuracy. Without adjusting the type of regularization technique or tuning the regularization hyperparameters, it is difficult to conclude what may have resulted in such performance decline, but perhaps the simple DNN model structures are limited in their prediction capabilities at around 50% accuracy without further data preprocessing and/or structure changes.

As for the CNN models, the trend is clear that adding another set of convolution/pooling layers along with increasing the number of filters proportionately has a significant positive effect on model performance, as evidenced in the test accuracy comparison between Experiment 3 and 4 as well as that between Experiment 7 and 8. However, it should be cautioned that without proper regularization, a higher number of convolution/pooling layers leads to the model overfitting and over-training quite severely. In Experiment 4, the model took over 17% more time to train than in Experiment 3 even though the trainable parameters were reduced by more than a million. The huge discrepancy between training accuracy and validation accuracy confirms the undesirable overfitting issue of the model in Experiment 4. Once the regularization techniques were applied, this issue was successfully mitigated in Experiment 8 resulting in favorable qualities: a shorter training time and higher test accuracy than Experiment 7.

Out of the first 8 experiments, the CNN model in Experiment 8 was considered the “best” performing. It was therefore selected as the baseline to perform further tuning of hyperparameters. ML Model tuning and optimization itself is an extensively studied subject that draws on both theoretical knowledge and empirical experience. For this assignment, I have opted to try a simple fixed-interval adjustment of the regularization parameter of dropout rate in Experiment 9 and 10. The results in Table 1 suggest that the original dropout rate of 0.3 performs better than both a higher dropout rate of 0.5 in Experiment 9 or a lower dropout rate of 0.1 in

Experiment 10. Perhaps a more mature way of exploring the hyperparameter tuning is through a grid search or Bayesian optimization which may adjust the dropout rate gradually to each convolutional layer.

Lastly, useful insights were gleaned from a further analysis of pooled feature maps (figure 6) in Experiment 3. Against the original horse image data (figure 7), the feature maps mostly extracted the edge features of the horse along different parts of its body including its face, neck, back, belly, legs and tail by comparing the "lighted up" regions to the original image. The horse shadow was notably extracted too as an edge feature. In Experiment 8, the t-SNE graph in figure 8 visualized how the predicted test labels from the same class can mostly be mapped to its own 2D cluster with some overlaps. From which can be inferred that the more tightly packed and the less overlapping the clusters are, the better the model performs.

5. CONCLUSION

In comparing different DNN and CNN architectures and the incorporation of regularization techniques, it is concluded from this assignment that the CNN in general out-performs a DNN in image classification tasks. Owing to the dual action in a CNN of the convolution operation that enables it to learn image features and apply this learned feature detector anywhere in the full image, and the pooling operation that aggregates statistics of these features at various image locations to streamline the model and enable "translation invariance", a CNN is a more superior type of NN model for stationary image classification.

Facial recognition technology consists of four stages, which include face detection, alignment, representation (facial feature extraction), and classification (Hu et al., 2015). Typically in facial recognition systems, the main challenge with crucial importance is the feature

representation scheme used to extract features and retain the most important information to represent a user's biometric trait. As CNN exhibits strong advantage in feature extraction over other types of ML procedures due to its inherent design, pre-trained state-of-the-art CNN models can be adopted and combined with other classifiers in transfer learning to develop a robust way to classify biometrics data on a human face.

Appendix

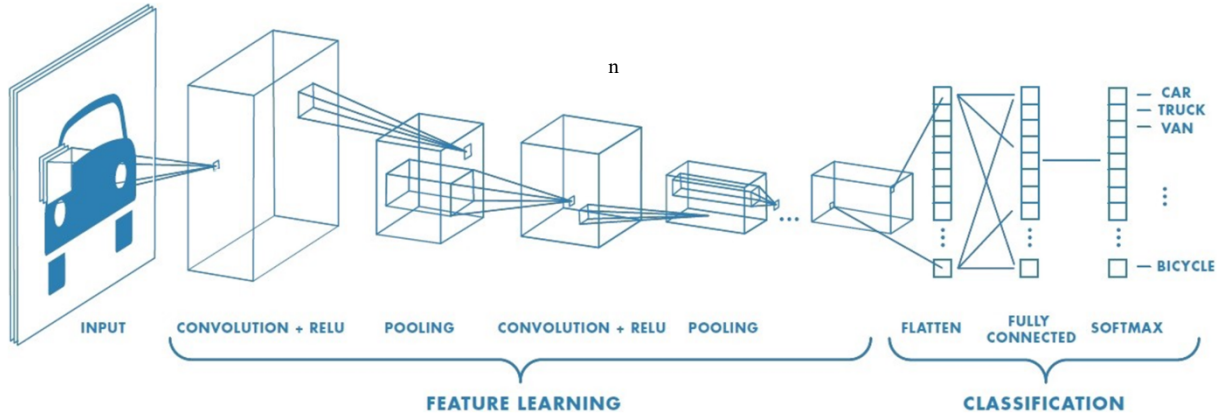

Figure 1. CNN Architecture (adapted from n https://github.com/djp840/MSDS_458_Public/blob/master/MSDS458_Assignment_02/MSDS458_Assignment_02_part01_v26_COLAB_20220906.ipynb).

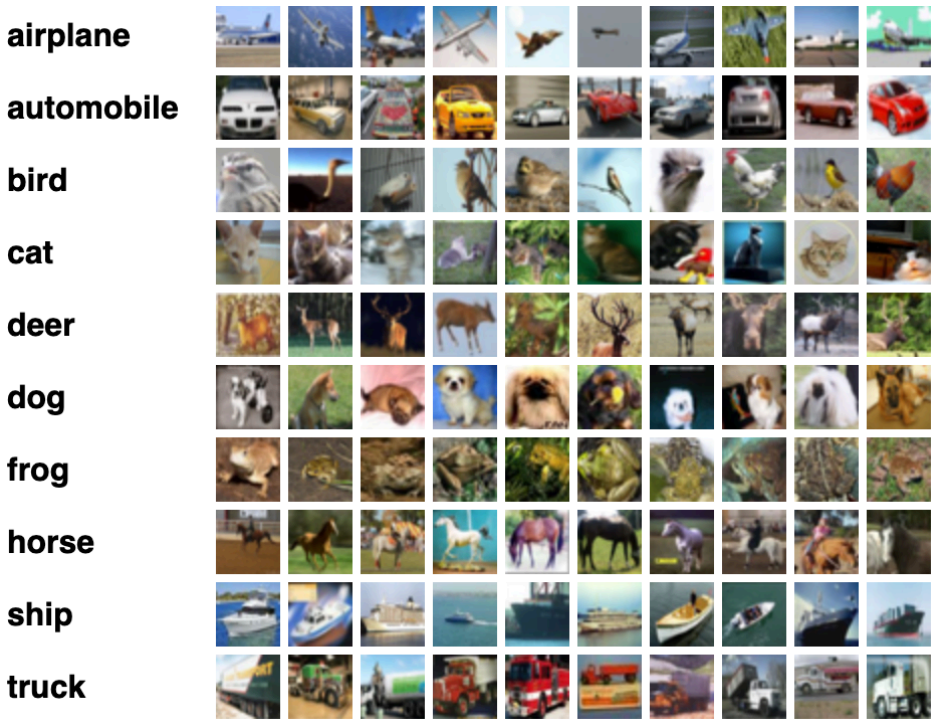

Figure 2. Sample CIFAR-10 images in all 10 classes (https://www.cs.toronto.edu/~kriz/cifar.html).

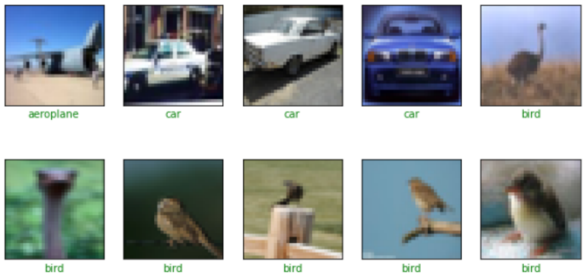

Figure 3. EDA: Sample CIFAR-10 images imported to Experiment 1.

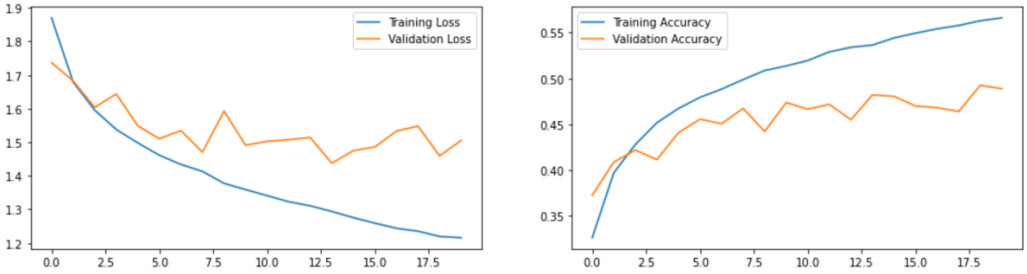

Figure 4. Model training/validation loss and accuracy history in Experiment 1

```
313/313 [==============================] - 1s 2ms/step
Classification Report
              precision    recall  f1-score   support

           0       0.48      0.61      0.54      1000
           1       0.54      0.69      0.60      1000
           2       0.40      0.39      0.39      1000
           3       0.35      0.32      0.33      1000
           4       0.48      0.34      0.39      1000
           5       0.45      0.30      0.36      1000
           6       0.46      0.60      0.52      1000
           7       0.65      0.46      0.54      1000
           8       0.61      0.63      0.62      1000
           9       0.52      0.59      0.55      1000

    accuracy                           0.49     10000
   macro avg       0.49      0.49      0.48     10000
weighted avg       0.49      0.49      0.48     10000

Accuracy Score: 0.4918
Root Mean Square Error: 3.15908214518078
```

Figure 5. (left) Classification report in Experiment 1. (right) Confusion matrix in Experiment 1.

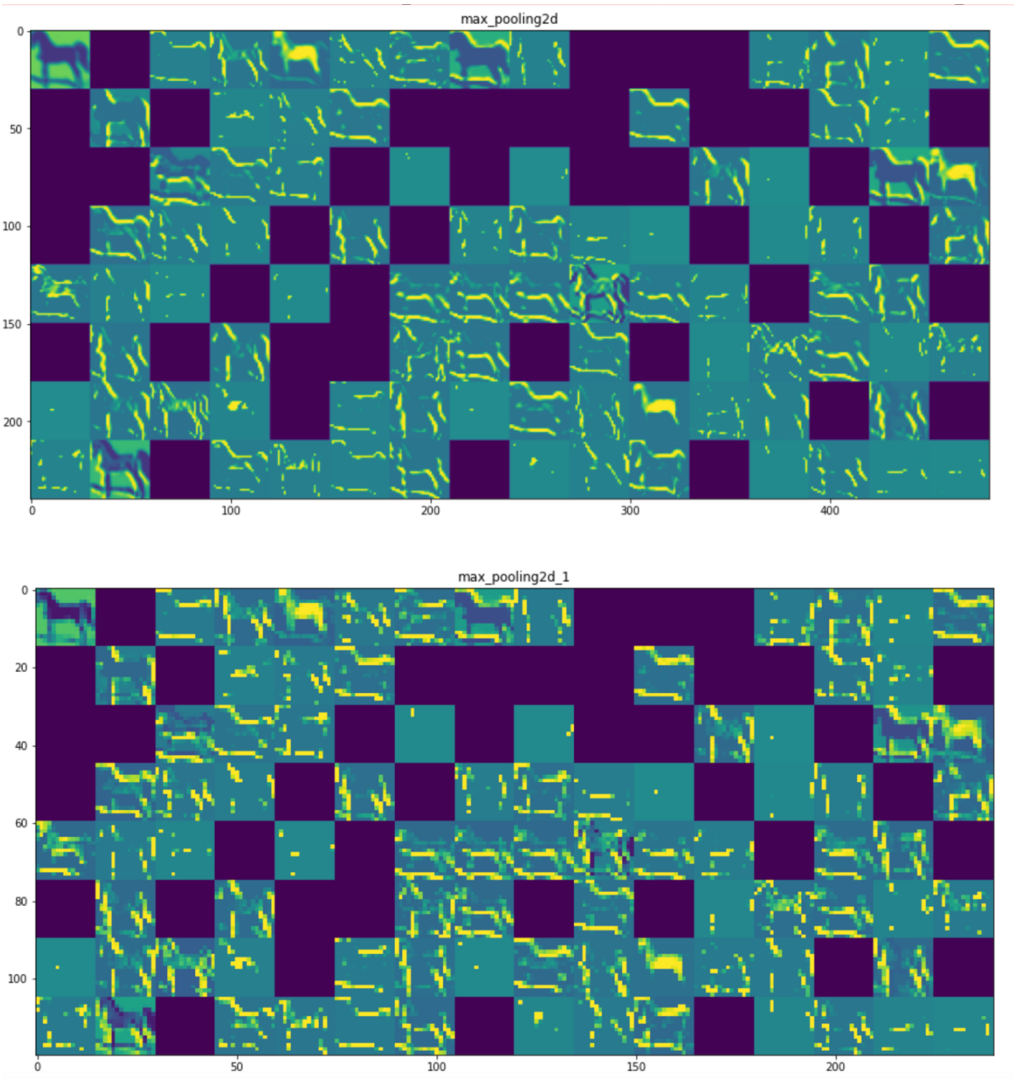

Figure 6. (top) Feature map grid of 1st max pooling layer in Experiment 3. (bottom) Feature map grid of 2nd max pooling layer in Experiment 3.

Figure 7. “Horse” sample image for testing feature extraction in Experiment 3.

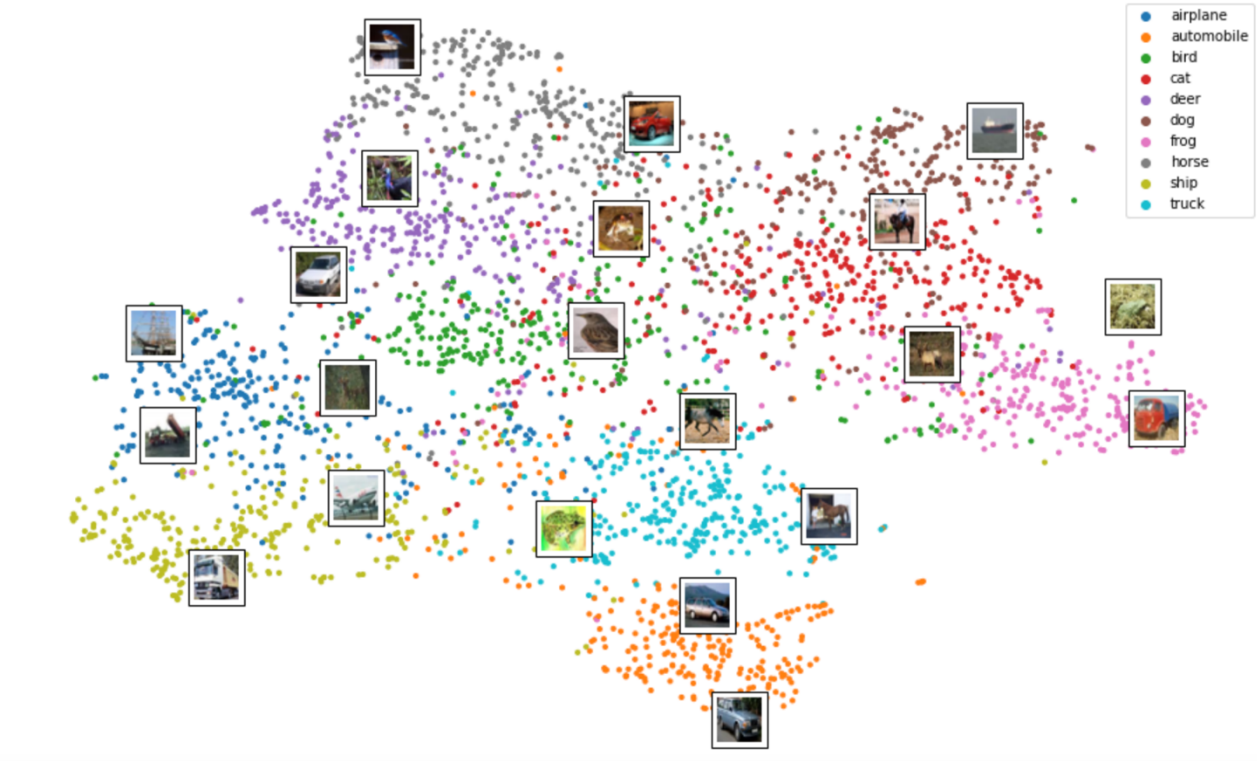

Figure 8. t-SNE graph of Experiment 8 test data predictions

	Model	Regularization	Dropout Rate	Epochs	Trainable parameters	Time to train	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy	Test loss
Exp 1	DNN 2 FC layers	No	-	20	1,331,722	44.06 s	56.6%	1.215	48.9%	1.505	49.2%	1.473
Exp 2	DNN 3 FC layers	No	-	20	1,479,562	46.068 s	64.2%	0.989	49.1%	1.627	49.1%	1.594
Exp 3	CNN 2 conv/pooling layers	No	-	20	3,841,930	154.199 s	70.4%	0.769	54.2%	1.305	53.5%	1.330
Exp 4	CNN 3 conv/pooling layers	No	-	20	2,269,578	177.242 s	83.2%	0.506	58.5%	1.300	59.0%	1.338
Exp 5	DNN 2 FC layers	Yes (Dropout, batch normalization, early stopping)	0.3	5	1,333,258	19.025 s	42.0%	1.635	41.4%	1.592	43.4%	1.559
Exp 6	DNN 3 FC layers	Yes (Dropout, batch normalization, early stopping)	0.3	12	1,481,866	51.072 s	43.8%	1.570	41.4%	1.612	47.5%	1.461
Exp 7	CNN 2 conv/pooling layers	Yes (Dropout, batch normalization, L2 kernel regularizer, early stopping)	0.3	17	3,842,698	149.191 s	74.8%	1.040	68.5%	1.224	73.6%	1.058
Exp 8	CNN 3 conv/pooling layers	Yes (Dropout, batch normalization, L2 kernel regularizer, early stopping)	0.3	13	2,270,346	128.167 s	77.8%	0.759	77.3%	0.783	76.7%	0.802
Exp 9	CNN 3 conv/pooling layers	Yes (Dropout, batch normalization, L2 kernel regularizer, early stopping)	0.5	21	2,270,346	205.276 s	70.4%	1.043	74.2%	0.925	74.5%	0.926
Exp 10	CNN 3 conv/pooling layers	Yes (Dropout, batch normalization, L2 kernel regularizer, early stopping)	0.1	17	2,270,346	163.222 s	89.7%	0.386	76.7%	0.878	75.9%	0.792

Table 1. Comparison of models in Experiment 1-10.

References

Laurens van der Maaten, & Geoffrey E. Hinton. (2007, December 31). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(86), 2579–2605. http://isplab.tudelft.nl/sites/default/files/vandermaaten08a.pdf

An Intuitive Explanation of Convolutional Neural Networks. (2017, May 29). Ujjwal Karn. Retrieved October 23, 2022, from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Chollet, F. (2021, December 21). *Deep Learning with Python, Second Edition* (2nd ed.). Manning.

Deshpande, A. (n.d.). *A Beginner's Guide To Understanding Convolutional Neural Networks*. Retrieved October 23, 2022, from https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

Alex Krizhevsky, Ilya Sutskever, & Geoffrey E. Hinton. (2012, December 2). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems, 25, 1097–1105. http://books.nips.cc/papers/files/nips25/NIPS2012_0534.pdf

Liu, R., Lehman, J., Molino, P., Such, F. P., Frank, E., & Sergeev, A. (2018, July 10). An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. Uber Blog. Retrieved October 23, 2022, from https://www.uber.com/en-NL/blog/coordconv/

Unsupervised Feature Learning and Deep Learning Tutorial. (n.d.). Retrieved October 23, 2022, from http://ufldl.stanford.edu/tutorial/supervised/Pooling/

Hu, G., Yang, Y., Yi, D., Kittler, J., Christmas, W., Li, S. Z., & Hospedales, T. (2015, December). When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition. *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. https://doi.org/10.1109/iccvw.2015.58