Evaluating Basic Convolutional Neural Network Structures

Coursework for Data Science 458: Research Programming Assignment II

Researcher: Michael Rocchio

Graduate Student at Northwestern University

Researcher Contact Email: MichaelRocchio2023@u.northwestern.edu

Abstract

In this research we will be looking at and comparing CNNs(Convolutional Neural Networks) to DNNs(Direct Neural Networks) in the sphere of image analysis and classification. Additionally, we will also be looking at some newer transformer models and discussing why they are better than CNNs.

Research, Dataset Introduction, & Literature Review

The purpose of the research below will be to, in 5 separate experiments, build and evaluate several different network architectures for the purpose of image classification on the cifar10 data set. Examples from this set are shown in **Figure 1**. Each image consists of 3072 pixels laid out on a 32x32x3 plane in RGB format, this is why there are 3 layers to each 32x32 image. These images are then converted into 3 dimensional NumPy arrays consisting of int values between 0 – 255 representing the intensity of the various pixels within greyscale image. The image set contains 60,000 images labeled between 0 and 9 which will be split into a train and test set consisting of 50,000 & 10,000 images, respectively. For the input into our direct neural networks we will be breaking down the 32x32x3 images into a single row of 3072 pixels. For CNNs and Transformers we will keep the images in the 32x32x3 format.

As there is no data leakage between the models, we will be using the test set as the validation data and evaluate our models based on that. We will also be min max scaling the pixels of the images by subtracting the min, 0, and dividing by the max, 255. We will not do that for the transformers. For each of the experiments to ensure that we control variables, we will be using the following parameters to compile the models; for an optimizer we will be using RMSprop, a standard use adaptive learning rate optimizer. To compute loss, we will be using sparse categorical cross-entropy as the y labels are not hot coded. The metrics observed will be categorical accuracy of the training and validation set. We will also be fitting the model with 120 epochs as the standard without early cutoff to achieve the goal of controlling the variables of the experiments. This will not be true for the best model and the transformers as I will be using various methods to optimize their performance.

Experiment I

In the first experiment we will be creating a 3-layer Dense Neural Network using Keras. The purpose of this will be to demonstrate in inefficacy of dense layers within networks containing only layer for comparison to the CNN models. The architecture of the network is as follows and can be visualized in **Figure 2**. The input layer will be incorporated into the hidden layer and contain 3072 input notes which is tailored to restructured cifar10 data. The one hidden layer will contain 1000 node activated by the Rectified Linear Unit function (RELU). This is the most utilized function within the hidden layers of neural networks as it is a simple and efficient linear function that doesn't overuse computing resources. Finally, the output layer will contain 10 nodes with SoftMax activation functions as we are doing a multi-classification output. After training the first model, the final validation accuracy was approx. 49% which was to be expected, a graph of the history of model_1's training metrics is shown

in **Figure 3**. This means that the model is unable to accurately predict the digits which is demonstrated in **Figure 4.**

Experiment II

In the second experiment we will be recreating all the same scenarios of the first experiment with the exception that there will be two hidden layers. This will very slightly increase the accuracy of the network but will not be sufficient in accurate classification of the images. A visual architecture of the network created in this experiment is demonstrated in **Figure 5**. Upon training the model we were able to achieve a validation accuracy of approx. 0.49%, which is to be expected as the second model; despite doubling the number of layers did not make it large enough to accurately predict the images as a taller DNN could achieve higher accuracy. This model also has been, due to the lack of regularization, severely overtrained as it has begun to pick up noise within the training set. Selected outputs for the model predictions are in **Figure 6.**

Experiment III & IV

During these experiments we will be creating our first CNNs(convolutional neural networks). The purpose of a CNN is to separate images into smaller squares(convolutions) in order to identify features unique to specific classes. An example of this is shown in **Figure 7**. It gives a breakdown of how a convolutional layer breaks down an image. In this case we are creating a 64-node output layer. This also demonstrates the greatest weakness of CNNs. They do not take into account the relative location of each kernel to it's surrounding kernels. During these experiments we did not utilize any regularization techniques which like the previous DNNs lead to severe overtraining of the model and thus a large divergence between

the accuracy of the training and test datasets. Additionally, an output from the request result 2 is shown in **Figures 8 & 9**, shows the regions that had the highest weight and thus should correspond to unique features that directly distinguish specific classes from one and another.

Experiment V

This experiment involves recreating all the previous experiments with the inclusion of various regularization which help prevent the model from picking up noise within the training set. One of the most common techniques is to create a max weight value thus evening out the weights of output nodes. This again decreases the models propensity to pick up irrelevant noise within the training set. After completing the experiments, we were able to drastically reduce the overtraining of the models and achieve an accuracy, set by my 'best model' of 60.60% on the validation data. I was however, after many iterations, not able to achieve a higher accuracy rate. I believe, after adding my own experiment, that this is due to the limitations of CNNs.

Experiment VI

During our final experiment I decided to show, quite painstakingly, show off the capabilities of visual transformers which allow the network to pick up the relative location of features within the kernels of split up images. ViTs address many of the weaknesses of CNNs and are, from personal experience, the go to for image classification today. An example of how a transformer breaks down images, similar to a CNN, is shown in **Figure 10.** After numerous iterations I was able to create a model that had a Training Accuracy of 88.93% and a Test Accuracy of 84.14%. This shows a model that is not overtrained and could the first prod model created during this research analysis thus far. A sample of the classification of the

images from the ViT is shown in **Figure 11**. The primary drawback, however, with ViTs is they take a massive amount of processing power and time to train.

Conclusion & Management Problem

While doing this assignment a lot was learned about the shortcomings of the once dominant CNNs for image classification. We have demonstrated why ViTs are the future of image classification and why many organizations are replacing their CNN models.

If I were to build a facial recognition algorithm for the users of a smartphone, I would consider a number of factors. First, I would be curious as to the input data. Many phones today such as the overpriced and underpowered iPhone employ this to good effect. However if we were to employ this using only simple images I would never employ a CNN as it would either be very unreliable or very insecure. Perhaps a ViT would work in this situation but that would be not great as well, the accuracy just wouldn't be there.

Literature Review

During my analysis, I spent a lot of time reading about CNNs and their shortcomings on medium's Towards Data Science publication until I came across a link espousing the greatness of visual transformers. So I went to Keras's website and decided to apply it to this current project. These articles were nothing but correct, CNNs are simply outdated and do not have a place in image classification as of about a year ago.

Bibliography

Sources used for coding:

https://keras.io/

https://www.tensorflow.org/

Used for coding result 2:

https://github.com/djp840/MSDS_458_Public/blob/master/msds458_assignment_02/MSDS458_Assignment_02_20210619_v12.ipynb

Research:

https://towardsdatascience.com/are-you-ready-for-vision-transformer-vit-c9e11862c539

https://towardsdatascience.com/is-this-the-end-for-convolutional-neural-networks-6f944dccc2e9

https://medium.com/@nabil.madali/an-all-mlp-architecture-for-vision-7e7e1270fd33

Figure Index

Model_Label	Training_Accuracy	Test_Accuracy	Train_Time
Experiment_1	0.6049	0.4919	384
Experiment_2	0.7076	0.4914	626
Experiment_3	0.9721	0.5959	645
Experiment_4	0.9508	0.5745	637
Experiment_51	0.3380	0.4007	498
Experiment_52	0.4265	0.4814	512
Experiment_53	0.8728	0.5623	782
Experiment_54	0.7814	0.5529	887
Experiment_55	0.5918	0.6060	957
Experiment_61	0.8926	0.8114	15240
Experiment_62	.8893	.8418	10728

Model Results Table (Training time is in seconds)

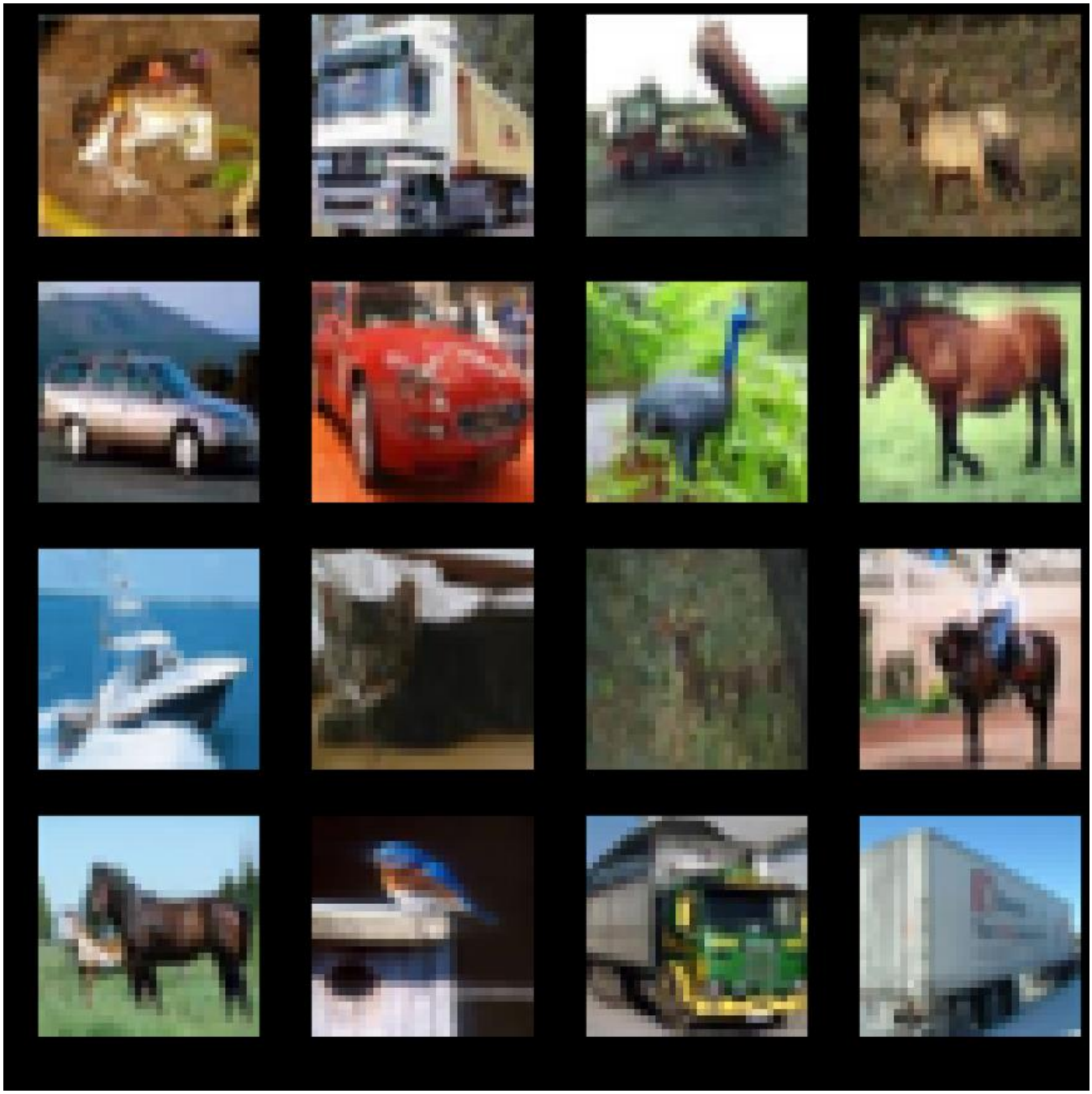

Figure 1. A Selection of 9 images from the MNIST dataset on 28 by 28 graphs.

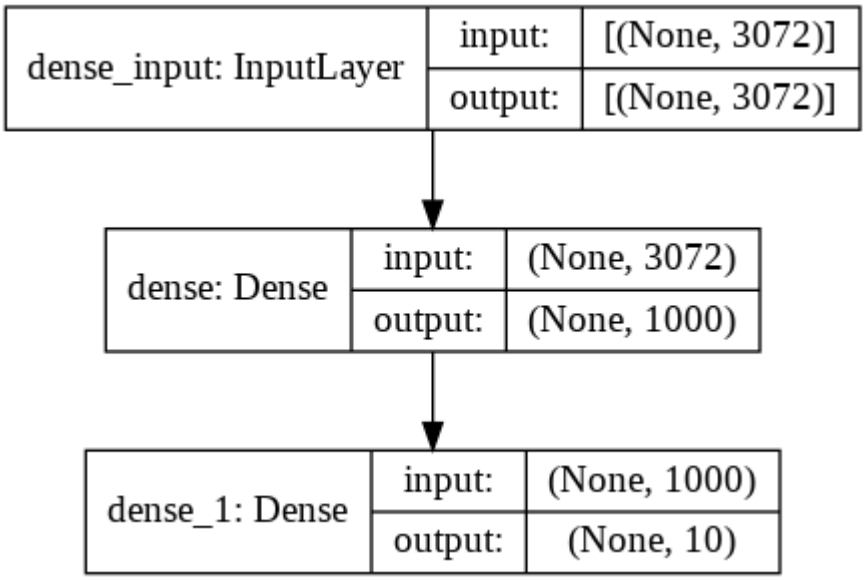

Figure 2. A visualization of the model created in Experiment 1.

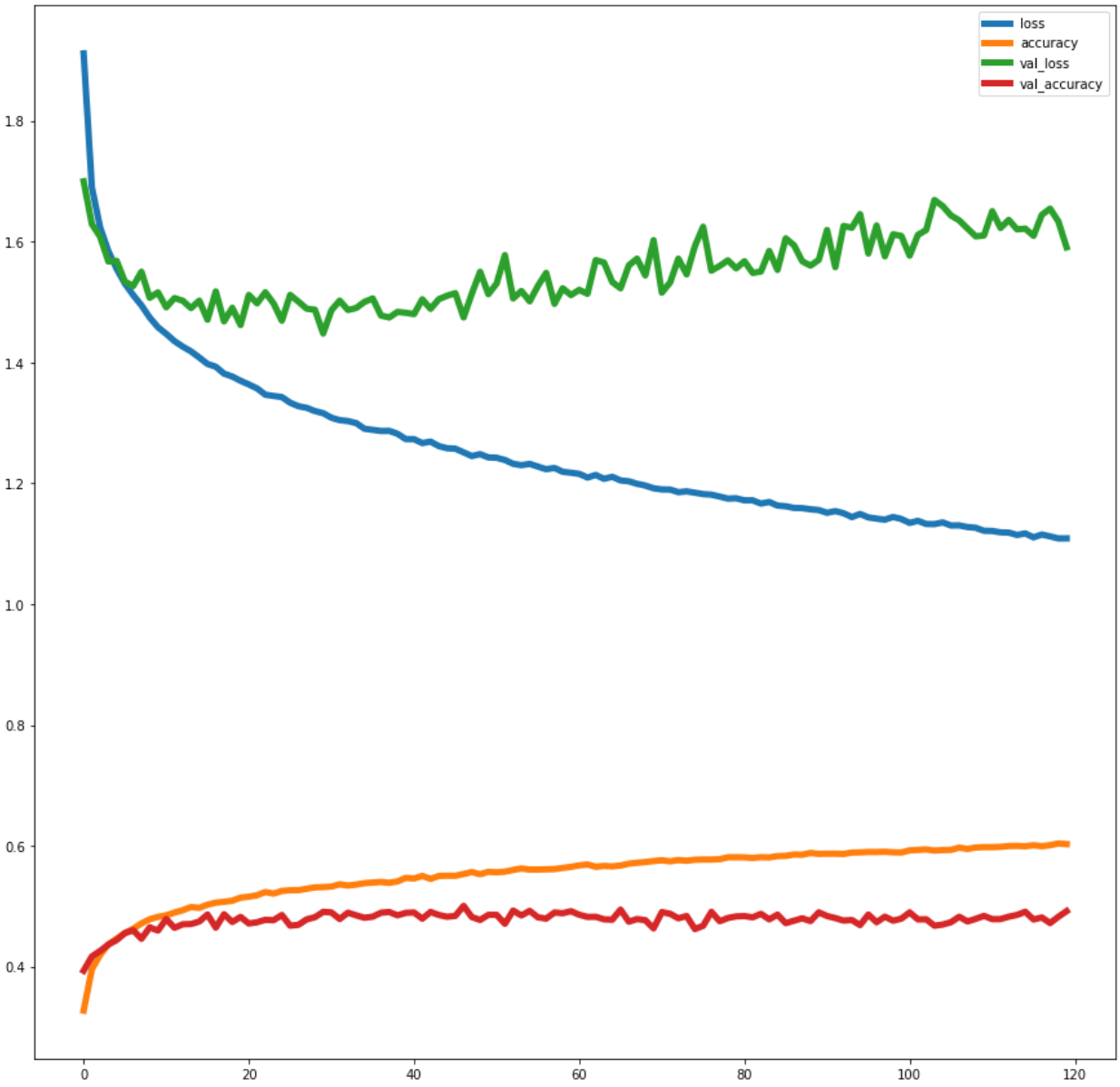

Figure 3. A historical graph of the metrics in model 1 over the 120 epochs.

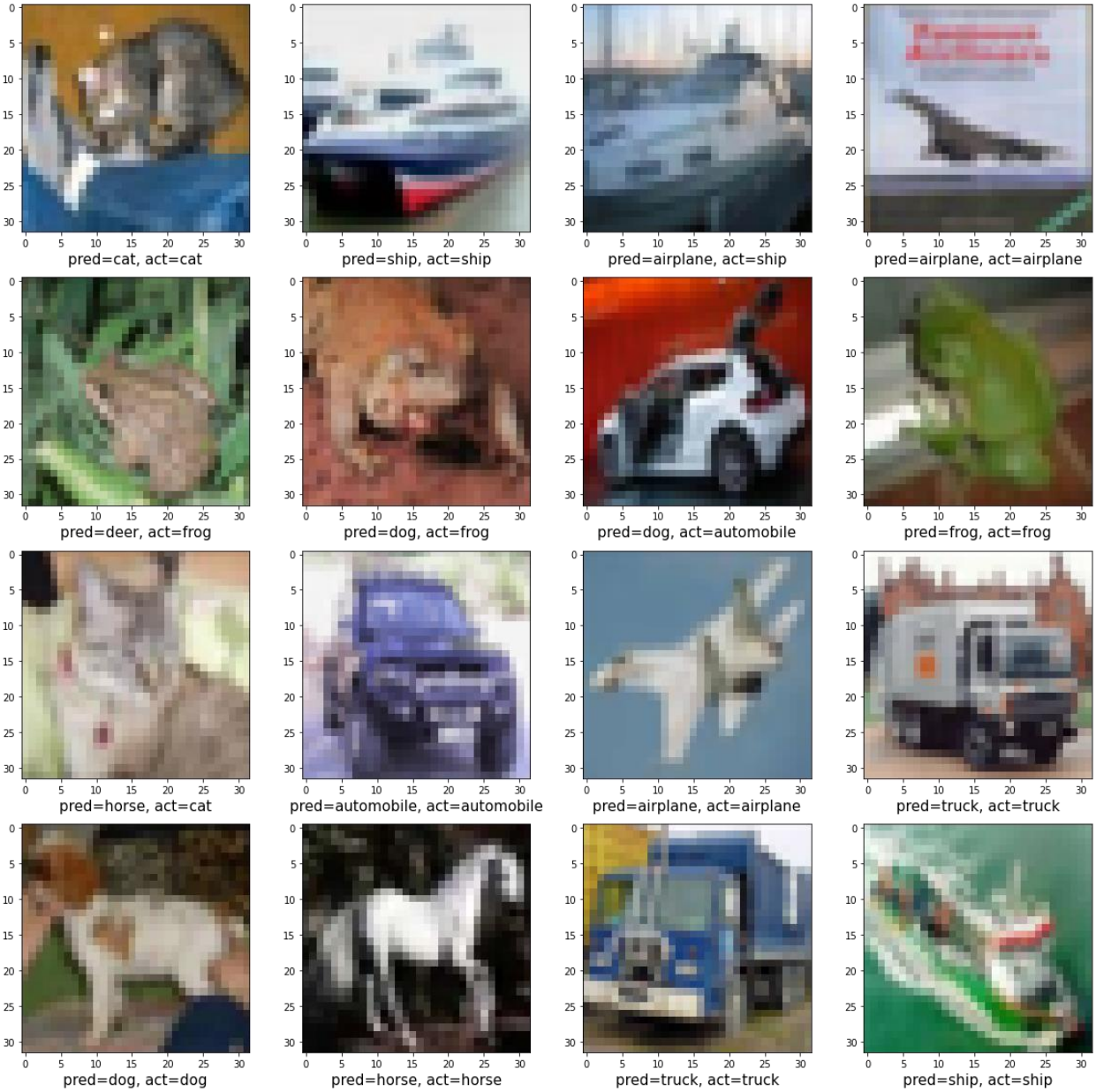

Figure 4. A random selection of 16 cifar10 images and their predictions from model 1.

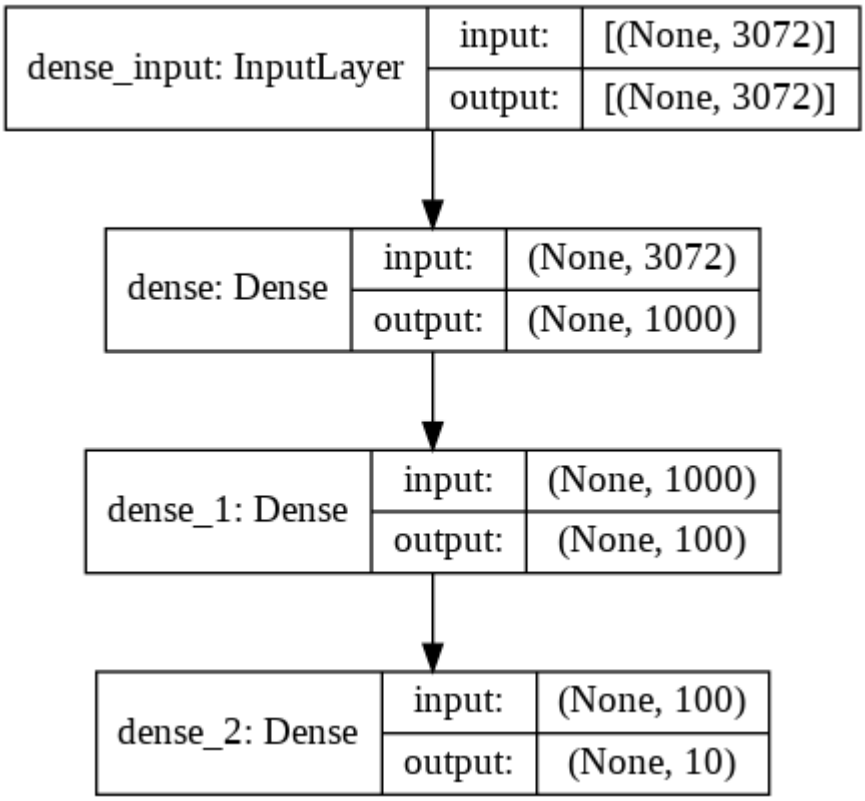

Figure 5. A visual architecture of the second model.

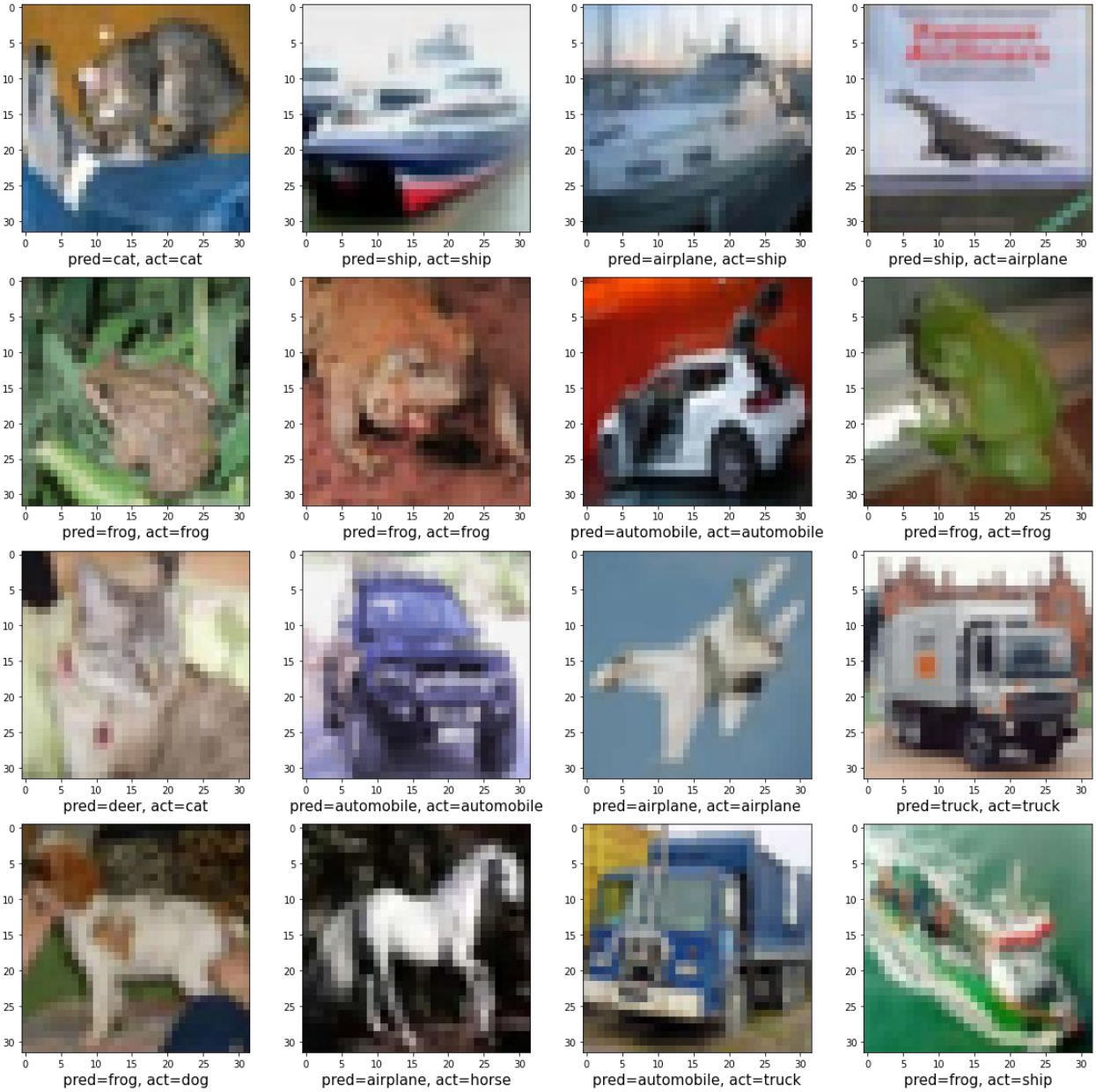

Figure 6. A random selection of 16 images and their predictions from model 2.

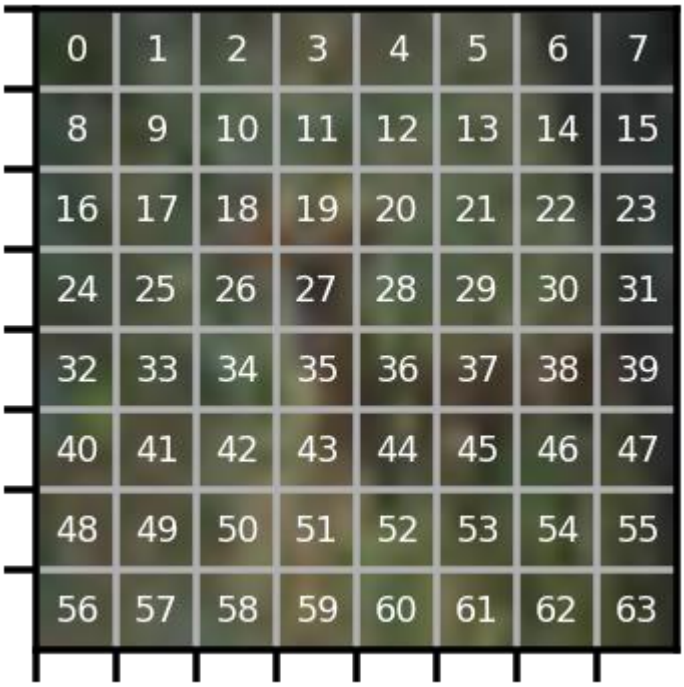

Figure 7. An example of how a convolutional layer splits up an image in our training set.

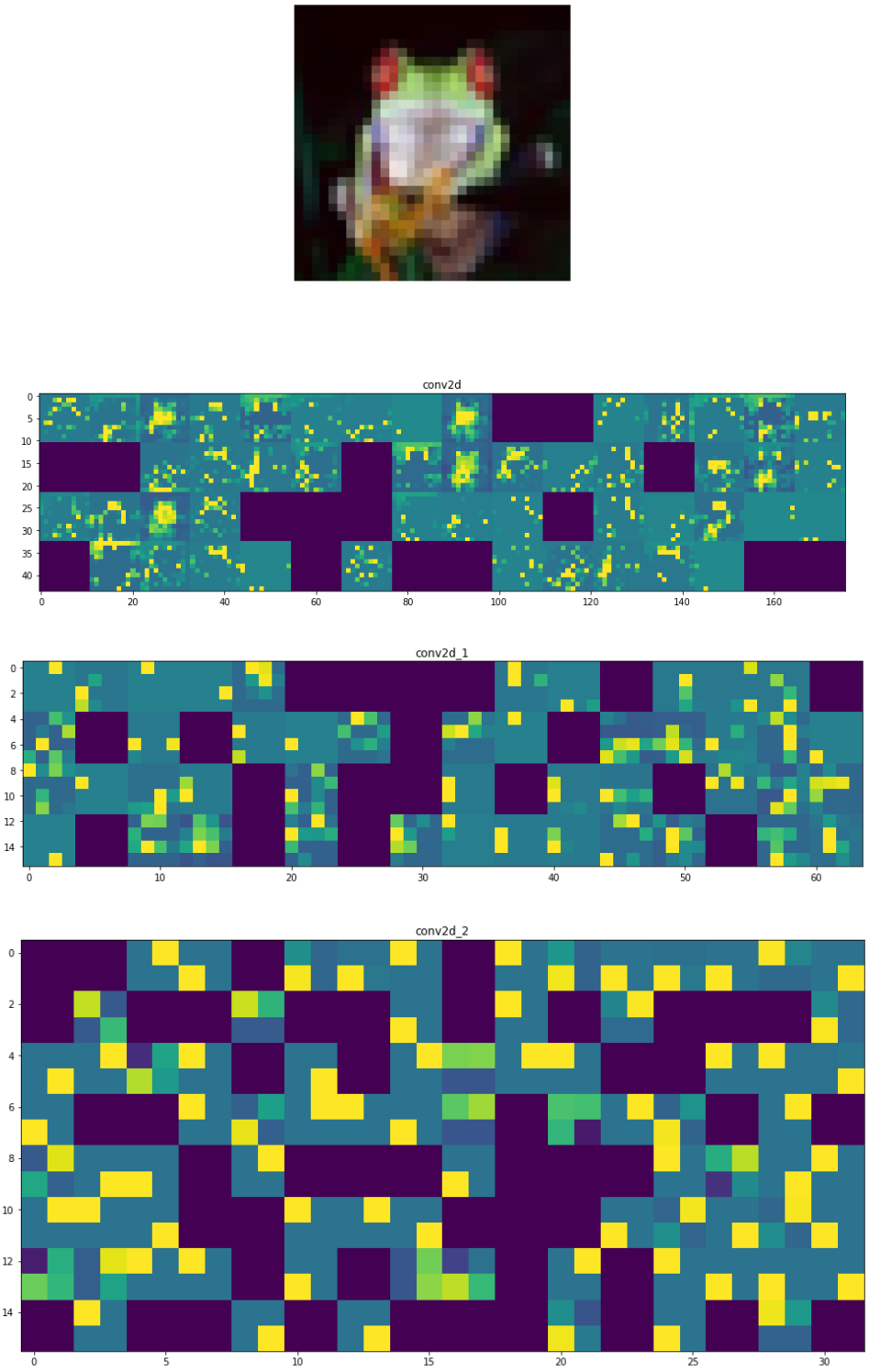

Figure 8. An example of the weighting placed on kernels in the convolutional net in model 3 at various layers. This shows a frog.

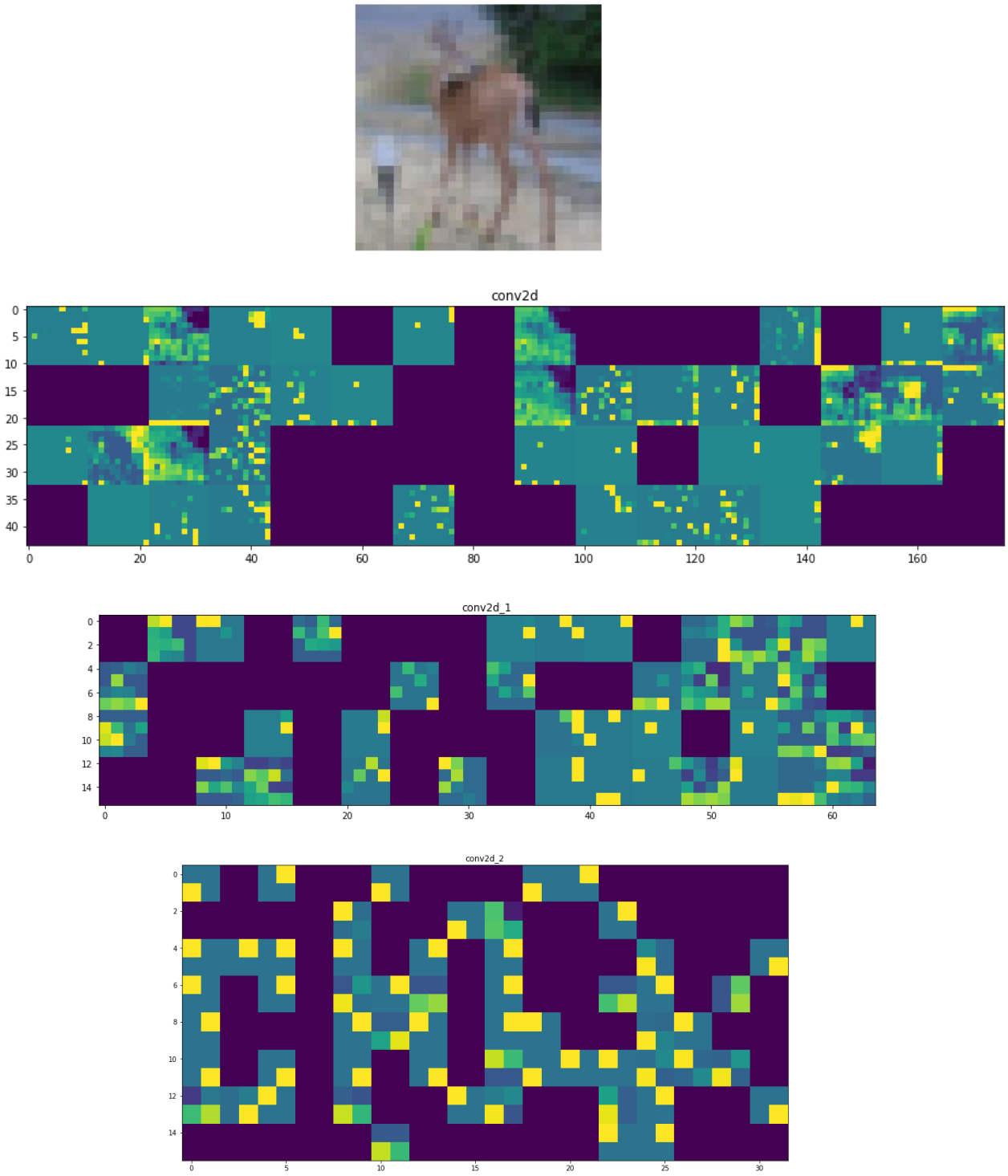

Figure 9. An example of the weighting placed on kernels in the convolutional net in model 3 at various layers. This shows a deer.

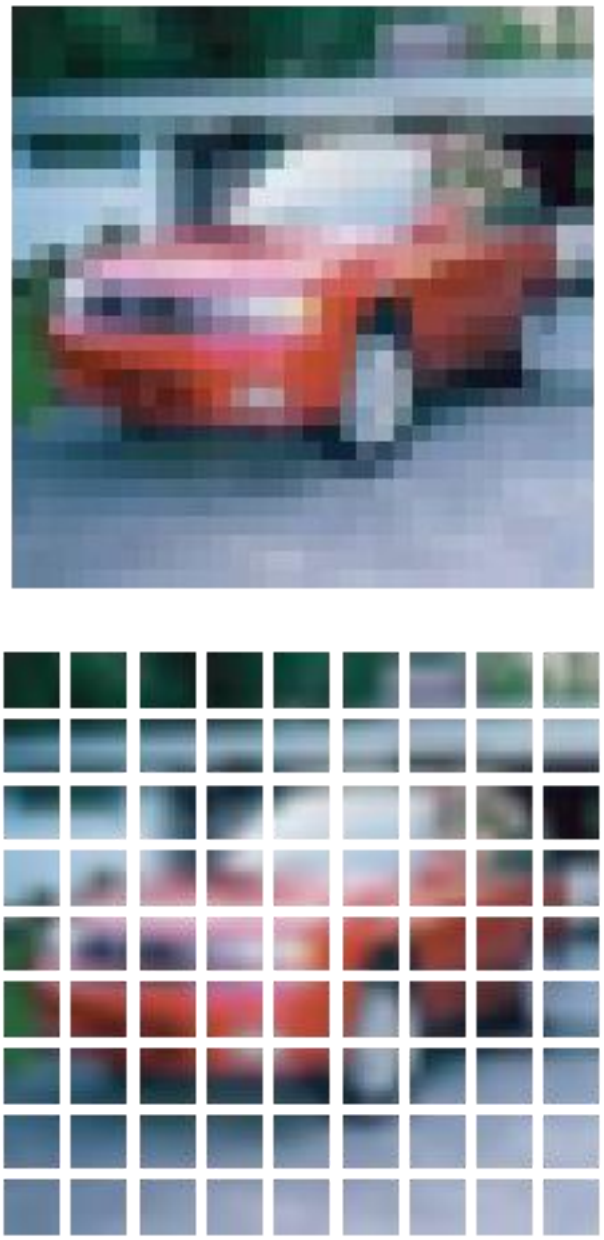

Figure 10. A example of how a ViT breaks down an image. This shows a car.

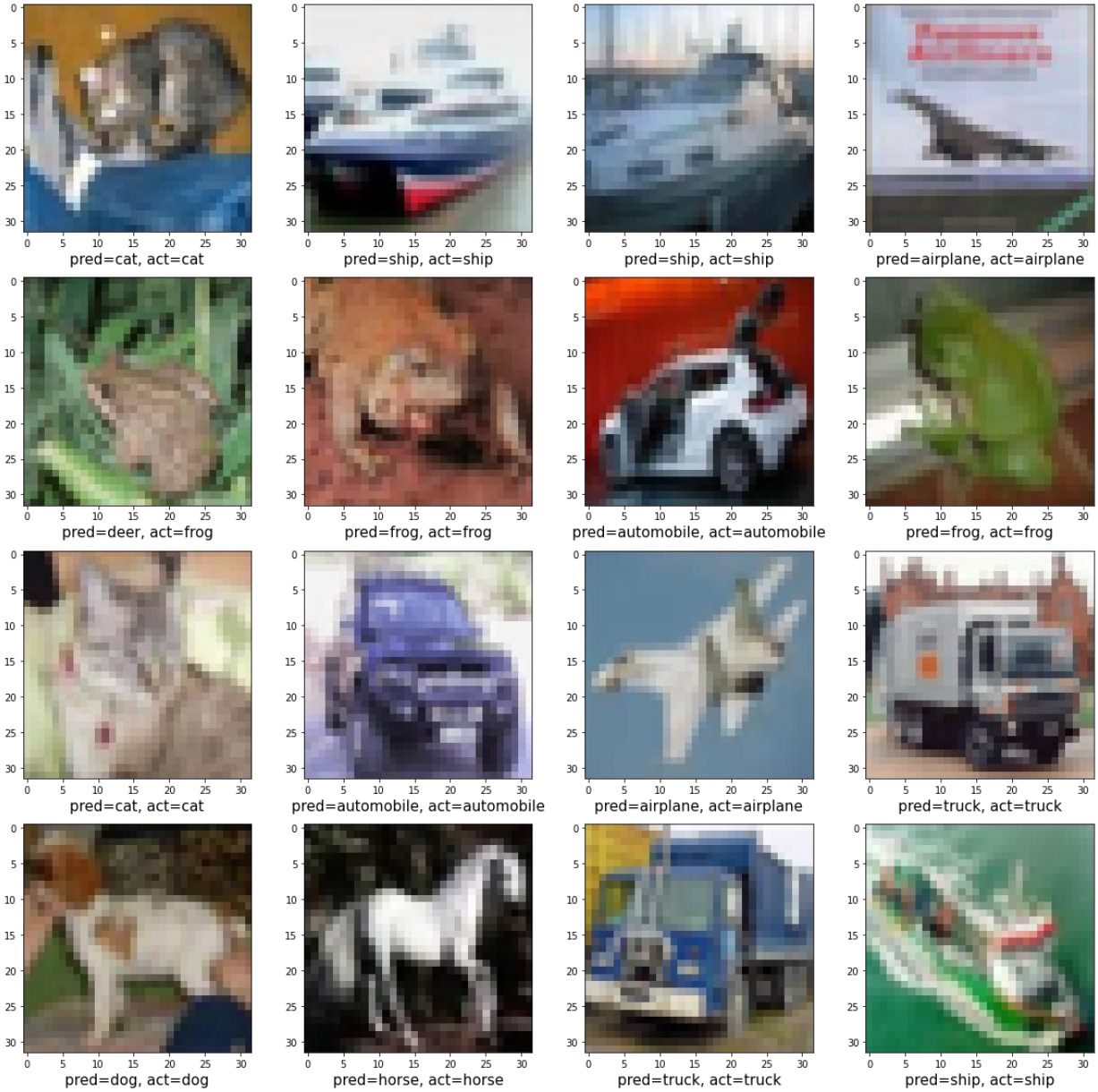

Figure 11. Selected output results from the final and best ViT model.