Project 3: Neural Networks

1st Michael Calderin
Department of Engineering Education
University of Florida
Gainesville, FL, USA
michaelcalderin@ufl.edu

***Abstract*—TensorFlow's CIFAR-10 dataset was used for image classification, providing 10 classes and 60,000 samples in total. In the first section, an artificial neural network was compared to the following classical models: random forest with dimensionality reduction and random forest without dimensionality reduction. With Python, the Gradio library was used to provide a user interface where pictures could be uploaded or drawn and the predicted class from the artifical neural network would be shown. In the second section, transfer learning with Google's vision transformer was used in an attempt to increase accuracy. Data augmentation, regularization techniques, and learning rate scheduling was explored.**

***Index Terms*—machine learning, transfer learning, CIFAR-10, neural network, artificial neural network, transformer, random forest.**

I. Introduction

CIFAR-10 is often used as a baseline or assessment for model performance on image classification. It includes the following classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. These correspond to labels 0 through 9, respectively, and the images are RGB with dimensions of 32x32 pixels. As a general measure of performance, [1] states that the human error rate on CIFAR-10 is about 6%, so models with above 94% accuracy generally perform better than humans. It is important to point out that the dataset is not purely clean. According to [1], the test set is estimated to have about 0.54% label errors and there are also some duplicates or near duplicates found across the training and testing sets.

In terms of modeling, vision transformers (ViT) and convolutional neural networks (CNNs) have become popular choices for image classification. Artificial neural networks (ANNs) and classical models such as random forest are typically less suited for this task due to their architectural inability to develop relationships between pixels and their color channels.

From a previous project of image classification using CIFAR-10 and classical models (logistic regression, random forest, and support vector machine), the best models with and without dimensionality reduction were both random forest classifiers. The optimal dimensionality reduction technique was principal component analysis (PCA) which out-competed manifold learning for the limited hyperparameter scope. In the current project, those two models were compared to an ANN. After, transfer learning was used with Google's ViT base model which was pretrained on ImageNet-21k and fine-tuned on ImageNet 2012. It was downloaded from Hugging Face [2].

II. Section 1: ANN vs. Classical Models

A. ANN Design and Training

During preprocessing, the dataset was split into 80% for training and 20% for testing. Samples were reshaped from 1x32x32x3 to 1x3072, flattening the channels. The features were scaled by dividing by 255. Color channels (RGB) are naturally bounded from 0 to 255 so this scaling was done to convert the interval to be between 0 and 1 to help with convergence.

From the Keras tuner, random search was used with 200 trials, 10 epochs, and early stopping with a patience of 5 epochs (monitoring for validation loss). A validation split of 20% was used and optimizations were for validation accuracy. The number of neurons in each layer were randomly picked from {16, 32, 64, 128, 256, 512}. Activation functions were picked from {"relu", "selu", "leaky_relu"}. Dropout rate was picked from {0.0, 0.1, 0.2, 0.3, 0.4, 0.5}. The number of hidden layers was picked from {2, 3, 4, 5, 6}. The optimizer was either Adam or stochastic gradient descent. Sparse categorical crossentropy was chosen as the loss function. Learning rates were picked from $\{1 \text{ x } 10^{-5}, 1 \text{ x } 10^{-4}, 1 \text{ x } 10^{-3}\}$. Batch size was picked from {16, 32, 64, 128, 256}. The kernel initializer for "selu" activation was "lecun_normal", otherwise it was "he_normal"; these pairings have empirically been shown to work well. This hyperparameter tuning took 25.85 minutes and the best validation accuracy was 0.41.

The best architecture is shown in Fig. 1. The batch size was 16 and the optimizer was Adam with a learning rate of $1\text{x}10^{-4}$. Notice that there is typically a chain of dense layers, batch normalization to prevent internal covariate shifts and help with convergence, and dropout to prevent overfitting. The output layer is a dense layer with 10 neurons and a softmax activation function so that there could be probabilities for the 10 classes. In an attempt to find better convergence, this architecture was re-trained with 200 epochs and a patience of 10 for early stopping. Training time took 20.87 minutes and resulted in a validation accuracy of 0.53, validation loss of 1.35, training accuracy of 0.58, and training loss of 1.20.

Fig. 2 demonstrates the accuracy across epochs. The validation accuracy is more unstable than the training accuracy and begins rounding off while the training accuracy continued to increase. This gives some indication that overfitting was beginning to kick in and the model was struggling with generalization. On the full training set, the macro average F1

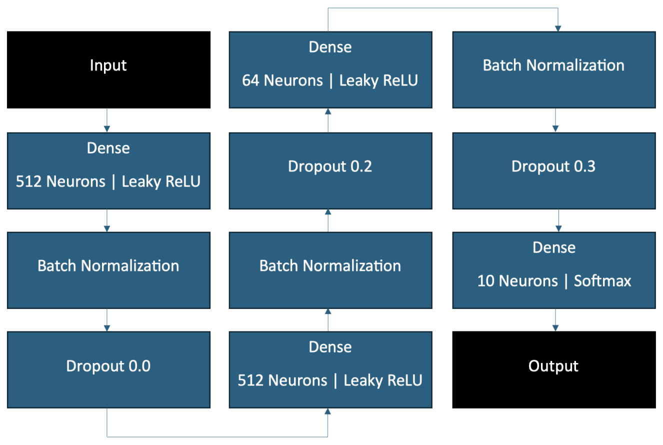


Fig. 1. ANN architecture after hyperparameter tuning using Keras tuner's random search.

score was 0.61 and accuracy was 0.61. These predictions took 2.42 seconds to execute. The automobile and ship classes had the highest F1 scores, both tied at 0.73. Cats had the lowest F1 score with 0.41.

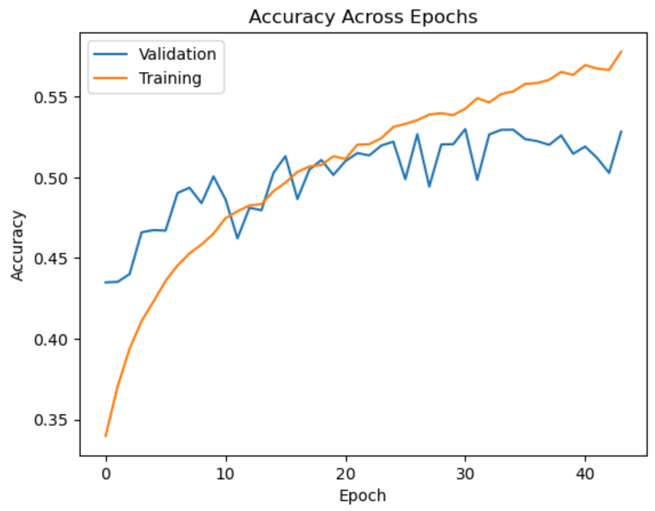


Fig. 2. Accuracy across epochs for the re-training of the ANN after tuning the hyperparameters.

The confusion matrix in Fig. 3 gives more insight into the struggling classes. Airplanes and ships were usually confused for each other, likely due to their shared blue backgrounds (ocean and sky). Automobiles and trucks were usually confused for each other which is expected since they are similar vehicles. The animal classifications are much more messy. For example, true cats are confused for dogs more often than the other classes but there are still significant confusions for the other animal classes such as frogs. Animals are typically much more unique with greater individual variability compared to vehicles. Their poses are also much less rigid, so this could help explain the difficulty with the animal classes.

In terms of the test set, it took 0.69 seconds to make the predictions and the results were a macro average F1 of 0.52 and accuracy of 0.53. The best and worst performing classes were no different than that of the training set. Fig. 4 shows the confusion matrix which has a striking resemblance to that of the training set and exhibits the same patterns. Based off these metrics, performance and generalization is quite weak for most tasks in a practical setting.

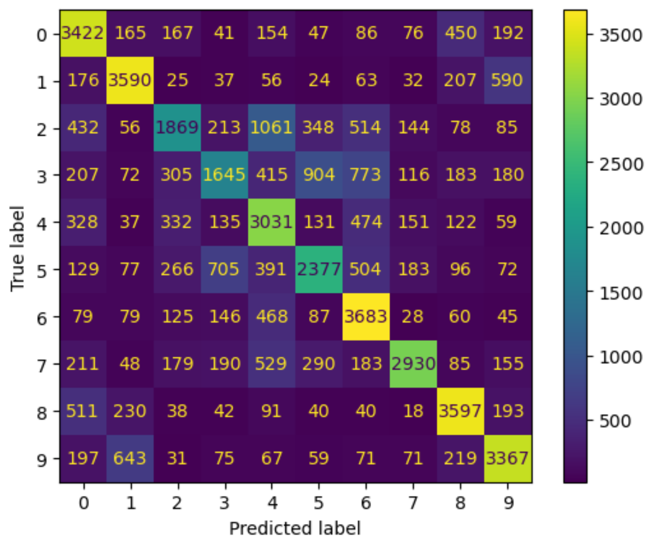


Fig. 3. Confusion matrix for ANN on the training set.

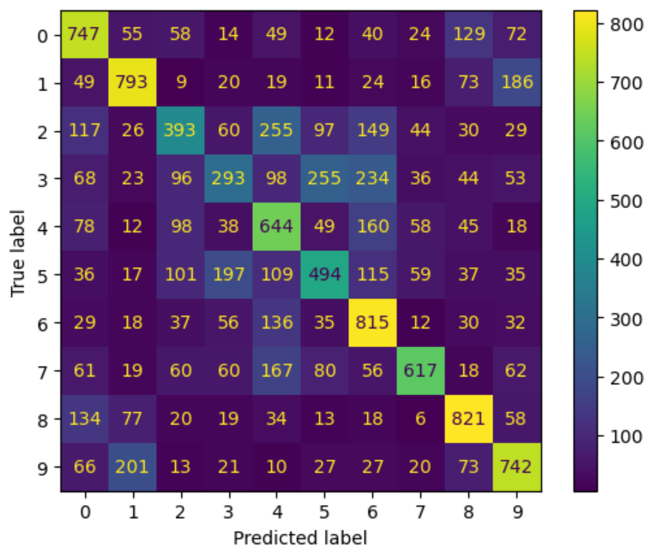
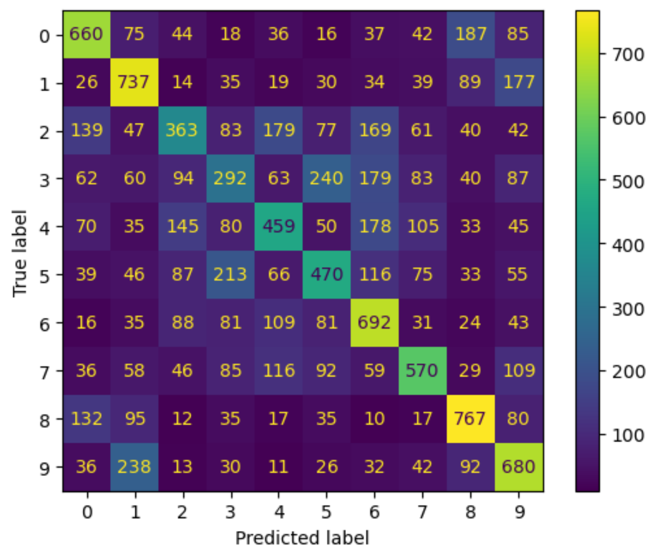


Fig. 4. Confusion matrix for ANN on the test set.

B. Comparison to Classical Models

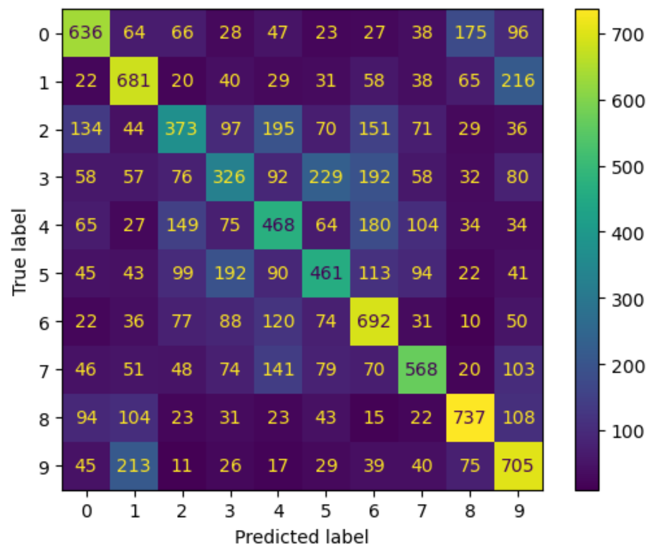
From a previous project using CIFAR-10 with classical models, the best dimensionality-reduced model was random forest with PCA and the best model without dimensionality reduction was also a random forest. For the current project, they were preprocessed the same way as the ANN and evaluated on the same splits. For the training set, both random forest models had a macro average F1 of 1.00 and accuracy of 1.00. Their confusion matrices were identical and had 4800 true positives for each class with no misclassifications. Predictions for the model without dimensionality reduction took 5.88 seconds and for the one with PCA they took 3.11 seconds.

For the test set, the random forest without dimensionality reduction had a macro average F1 of 0.47, accuracy of 0.47, and predictions took 1.52 seconds. Its confusion matrix is shown in Fig. 5. The random forest with PCA had a macro average F1 of 0.47, accuracy of 0.47, and predictions took 0.80 seconds. Its confusion matrix is shown in Fig. 6. For both models, the best F1 was for ships and the worst was for cats, similar to ANN. As seen in the confusion matrices, there is similar behavior to the ANN where the struggle in terms of generalization is confusing vehicles for each other and confusing animals for each other. It is probable that the higher proportion of animals to vehicles contributes to the more intense confusion among the animal classes, aside from animals being inherently more difficult to classify due to their higher variance.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	660	75	44	18	36	16	37	42	187	85
1	26	737	14	35	19	30	34	39	89	177
2	139	47	363	83	179	77	169	61	40	42
3	62	60	94	292	63	240	179	83	40	87
4	70	35	145	80	459	50	178	105	33	45
5	39	46	87	213	66	470	116	75	33	55
6	16	35	88	81	109	81	692	31	24	43
7	36	58	46	85	116	92	59	570	29	109
8	132	95	12	35	17	35	10	17	767	80
9	36	238	13	30	11	26	32	42	92	680

Fig. 6. Confusion matrix for random forest with PCA on the test set.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	636	64	66	28	47	23	27	38	175	96
1	22	681	20	40	29	31	58	38	65	216
2	134	44	373	97	195	70	151	71	29	36
3	58	57	76	326	92	229	192	58	32	80
4	65	27	149	75	468	64	180	104	34	34
5	45	43	99	192	90	461	113	94	22	41
6	22	36	77	88	120	74	692	31	10	50
7	46	51	48	74	141	79	70	568	20	103
8	94	104	23	31	23	43	15	22	737	108
9	45	213	11	26	17	29	39	40	75	705

Fig. 5. Confusion matrix for random forest without dimensionality reduction on the test set.

C. Performance Analysis

In terms of metrics on the test set, ANN performs the best. It has the highest accuracy and macro average F1 score with the fastest prediction time. The two random forest models share similar performance, but the PCA version outperforms the version without dimensionaltiy reduction since it has a simplified feature space, allowing for faster predictions with comparable accuracy and F1. Both random forest models show severe signs of overfitting while RNN has much less drastic overfitting issues.

Still, none of the models are doing well. For example, the highest test accuracy is 0.53% (for ANN) which is unacceptable for most applications. Neither ANN nor random forest are equipped to handle colored images. There are 3 color channels for a given pixel and there is no innate way in their infrastructure to associate them. Meaning, if an input sample were a 1x32x32x3 tensor then each pixel would be closely associated with its 3 color channels. However, all these models required a flattened 1x3072 input which loses the connection between a pixel and its channels.

All models have the same classification struggles: vehicles are confused for each other and animals are confused for each other. This has less to do with the architecture of the models and more to do with the nature of the classes. Even for a human, it would be more difficult to distinguish a cat from a dog compared to a cat from a car. The troublesome classes would likely be troublesome for a wide array of models. However, overfitting was seen more prominently in the random forest models over the ANN. Their trees likely grew too deep and began memorizing the training data which could have been from an inability to capture shape-based patterns. Dropout in the ANN was used to prevent overfitting and for the most part it seemed to work. The models are likely associating classes with similar levels of color channels, but are not actually understanding patterns in terms of shapes.

D. Gradio Inference Interface

Using Gradio, a library for Python, a user interface was built to handle real-time ANN predictions. Users could upload an image and have the predicted class be displayed in a text box. They could also draw an image and the text box would update in real-time with predictions as the drawing progressed. This was provided as a cell in a Jupyter Notebook and a demo was recorded to showcase the features.

III. Section 2: Using Transfer Learning

A. *Choosing and Implementing a Model*

The ViT main layer from [2] was used as the pretrained base. It was wrapped to be a subclass of a Keras layer so that it would be compatible with TensorFlow. It was then connected to a Dense output layer with 10 neurons and a softmax activation function to predict the 10 classes of CIFAR-10. The input required a shape of 3x224x224, where 3 represented the RGB channels and 224x224 was the required pixel dimensions for images. Hugging Face came with its own preprocessor which resized images to be 224x224 and normalized the RGB channels to have mean (0.5, 0.5, 0.5) and standard deviation (0.5, 0.5, 0.5).

The ViT layer was frozen and Adam optimizer was used with sparse categorical crossentropy as loss and a 0.001 learning rate. Due to the size of the training dataset and complexity of the model, the model was fit in batches and 4-fold cross-validation was used for each batch, stratifying based on the target label when splitting for both batches and folds. The train-test split was 50,000 to 10,000 samples which is the default split for CIFAR-10. The training set was divided into 50 batches but only 25 were trained on. This was required since resizing the images for the entire dataset took up more memory than available. For fitting, the batch size parameter in TensorFlow was left as 32, not to be confused with the previously discussed ”batches”. This architecture was regarded as the ”baseline” model which would give a reference when trying to make improvements.

Across all folds, the average training accuracy was 0.96 and the average validation accuracy was 0.97. The accuracy across epochs is shown in Fig. 7. Note, the passes were labeled as ”epochs”, but each epoch is actually a fold and not a full pass of the entire training dataset (only a pass of a fold). Notice, training and validation accuracy quickly saturated with few epochs and then flatlined, although with fluctuations, with very little improvement. Even with a lower learning rate, similar fluctuations were observed and accuracy did not increase; this is explored later when improvements were attempted.

Training took 186.47 minutes and predictions generated for each batch took around 95 seconds. For evaluation on the training set, performance was measured on the first and last batch used (1st and 25th batch). This was to ensure the model was not forgetting information from older batches and was also learning from the more recent batches. For the first batch, the accuracy was 0.97 and the macro average F1 score was 0.97. Its confusion matrix is shown in Fig. 8. For the last batch, the accuracy was 0.98 and the macro average F1 score was 0.99. Its confusion matrix is shown in Fig. 9. There was stronger performance and less misclassifications for the last batch compared to the first. It gives indication of forgetting, but performance was still strong for both batches. Confusion between cats and dogs seemed to be the most prominent. They are both domestic pets with similar figures and similar backgrounds so this is a reasonable mistake. Especially, since the images are low quality which sometimes takes away the finer details that differentiate the two.

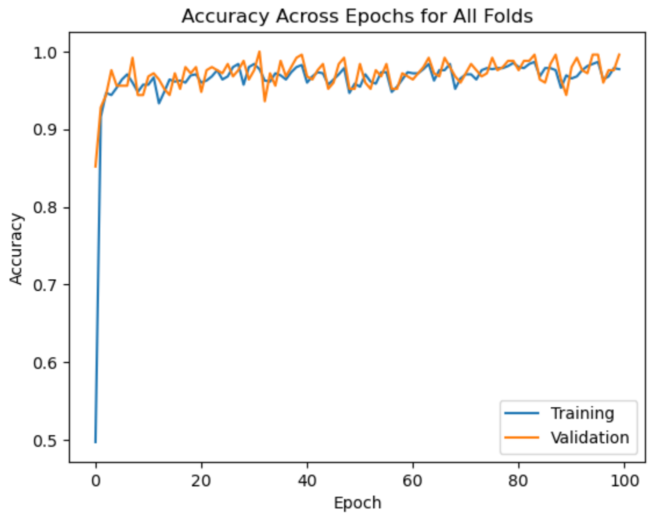
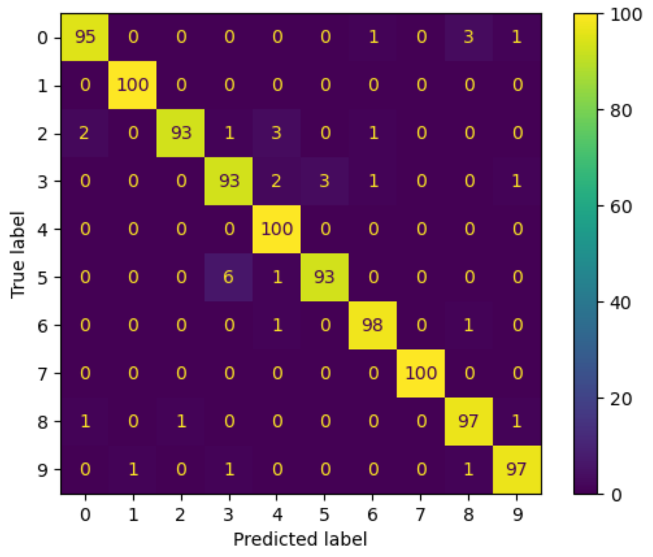


Fig. 7. Accuracy across epochs for baseline transformer model. Each ”epoch” is actually a fold and not a full pass of the entire training set.



	0	1	2	3	4	5	6	7	8	9
0	95	0	0	0	0	0	1	0	3	1
1	0	100	0	0	0	0	0	0	0	0
2	2	0	93	1	3	0	1	0	0	0
3	0	0	0	93	2	3	1	0	0	1
4	0	0	0	0	100	0	0	0	0	0
5	0	0	0	6	1	93	0	0	0	0
6	0	0	0	0	1	0	98	0	1	0
7	0	0	0	0	0	0	0	100	0	0
8	1	0	1	0	0	0	0	0	97	1
9	0	1	0	1	0	0	0	0	1	97

Fig. 8. Confusion matrix for baseline transformer on the first batch of training samples used.

Predictions for the full testing set took 22.55 minutes. The accuracy was 0.96 and macro average F1 was 0.96. The confusion matrix is shown in Fig. 10. The most prominent confusion is still between cats and dogs. Some less prominent yet significant misclassifications exist between classes such as airplanes and ships, and automobiles and trucks. These are similar mistakes to those seen previously with ANN and random forest. For low-stakes applications, the performance is quite strong and overfitting is not an issue. However, the weaker performance for classes such as cats and dogs would need to be noted or addressed.

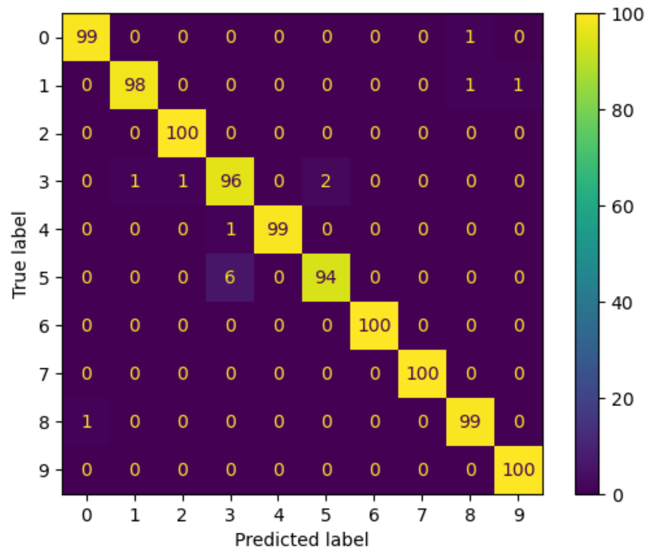


Fig. 9. Confusion matrix for baseline transformer on the last batch of training samples used.

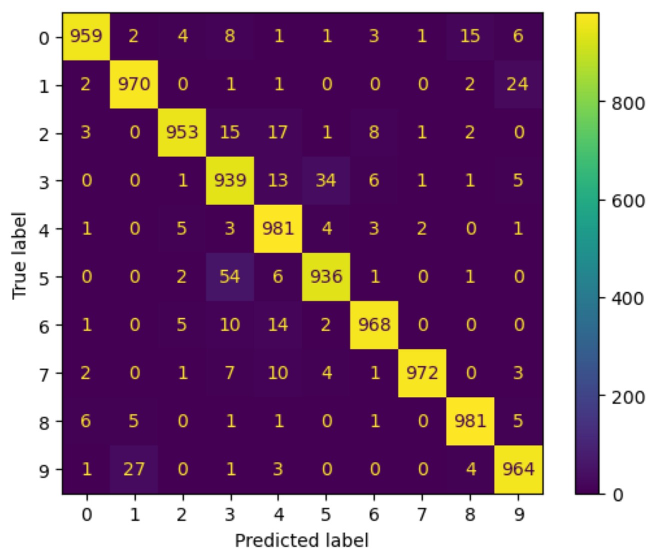


Fig. 10. Confusion matrix for baseline transformer on the test set.

B. Improvements

A fine-tuned instead of frozen configuration was not used due to computational and time constraints. There was also difficulty with making the layer trainable since adjusting the trainable parameter to "True" seemed to have no effect. Instead, attempts for improvement were in terms of data augmentation, regularization, and learning rate scheduling. For augmentation, 3 transformations were applied before the ViT and output layer: random horizontal flips, random rotations ($\pm72°$), and random zooms (in and out by 10%). The rest of the preprocessing and training steps align with the baseline model, except 2 out of the 50 batches were used for training. Less batches were used throughout the improvement process since it allowed for more configurations to be attempted.

For augmentation, training time took 16.59 minutes and predictions for a given batch took around 110 seconds. The average training accuracy across all folds was 0.66 and the average validation accuracy was 0.89. The training samples were randomly transformed but the validation samples were not so validation samples were easier to predict, likely attributing to the higher score. Note the validation samples were not transformed to be able to make direct comparisons to the baseline model. On the first training batch, accuracy was 0.95 and the macro average F1 was 0.95. On the last training batch, accuracy was 0.96 and the macro average F1 was 0.96. The behavior was similar to the baseline model where performance was slightly better for the most recently trained batch, and based on the confusion matrices, the troublesome classes were generally the same.

On the test set, augmentation took 22.53 minutes to generate predictions. Accuracy and macro average F1 were both 0.94. Its confusion matrix again showed that cats and dogs were the most commonly confused classes. Specifically, cats were confused for dogs more often than dogs were confused for cats.

The regularized model was similar to the baseline model but before the output layer, it had a Dense layer with 64 neurons, ReLU activation, the kernel regularizer set to L2 of 0.001, the kernel initializer as "he_normal", followed by dropout with a rate of 0.3. He Initialization with a normal distribution has been empirically shown to work well with ReLU, and the values such as 0.001 for L2 were populated based on typical ranges used in other experiments. 2 out of 50 folds were used and this resulted in a training time of 15.07 minutes and average training accuracy of 0.90 across the folds and average validation accuracy of 0.96. Predictions for each batch took around 100 seconds. The first batch had an accuracy and F1 (macro average) of 0.98. The last batch had an accuracy and F1 of 0.99. The test set predictions took 22.35 minutes and had an accuracy and F1 of 0.95. The same patterns for misclassifications were shown in the confusion matrices compared to the other models. The regularized and baseline models had less misclassifications for cats compared to the augmented model, but it was the class with the highest number of misclassifications across the board.

The augmented and regularized architecture was combined and trained with 2 batches. Training took 15.23 minutes and had an average training accuracy of 0.66 and average validation accuracy of 0.92 across the folds. Training predictions took about 100 seconds. The first batch had an accuracy and F1 of 0.95. The last batch had an accuracy of 0.96 and an F1 of 0.97. Predictions for the test set took 22.37 minutes and resulted in an accuracy of 0.95 and F1 of 0.95. Its confusion matrix was similar to that of the purely augmented model, especially in its weaker performance for the cat class.

In an attempt to address the noisy validation accuracy and validation loss of the baseline model, a harsh learning rate scheduler was used which began with a learning rate of 1 x 10^{-4} and reduced by a factor of 0.5 across each fold, with a minimum learning rate of 1 x 10^{-7}. Training continued where the baseline model left off. The training set was split into 50 batches but with a different random state so that the training samples would be different from the original ones used to train the baseline model. 4 out of 50 of those batches were used which took 40.56 minutes. The average training accuracy and validation accuracy was both 0.97. As the model was fed more data, validation accuracy showed signs of increasing but fluctuations were still present which made it difficult for convergence at a higher accuracy.

Predictions for each batch took around 140 seconds. The first batch had an accuracy and F1 of 0.97. The last batch had an accuracy and F1 of 0.96. This was the first time the first batch outperformed the last batch. Predictions for the test set took 22.31 minutes and resulted in an accuracy and F1 of 0.96. Its confusion matrix was similar to the baseline model, having extra trouble with dogs. The other models had the most misclassifications for cats while for this model and the baseline model, dogs were the most misclassified.

C. Interpretation

Referring to Fig. 7, the models gave similar indications that they would have behaved like the baseline model if given more epochs to train. Essentially, saturating to a high accuracy quickly and then fluctuating without improvements. Even with continuing to train the baseline model with a lower learning rate, it provided no improvement in this regard. It seems like once roughly 95-96% accuracy is reached, it is difficult to make any more improvements, at least with the current computational resources and methods.

Fig. 11 shows random samples of correct predictions from the best performing model, the baseline model. Some of the images were difficult to see, but the model surprisingly classified them correctly. For example, the lower-right cat is very blurry. There are a few examples with horses and the model seems to understand both their face and body well.

Fig. 12 shows some incorrect predictions from the baseline model. Some of these are obvious to a human, but others are incrdedibly difficult to make out. On the last column of the second row, the true label is ”horse” but it looks like some sort of coyote. This is likely a label error in the CIFAR-10 dataset. These factors could explain why it is difficult for the models to achieve higher than 96% accuracy. Recall that humans have an error rate of 6% for this dataset so the models are performing better than humans but are likely limited by the low quality images and label errors. It would be suspicious for a model to get, say 100% accuracy, knowing that inconsistencies exist in the actual target labels. The frozen layer could have also been a limitation, preventing that push over 96% accuracy. Fine-tuning with a lower learning rate and the full architecture unfrozen could have been the slight tweak needed.

D. Summary and Performance Comparison

Augmentation was used in an attempt to help with generalization. The goal was for the model to see transformed images and realize the classes are usually differentiated by abstract

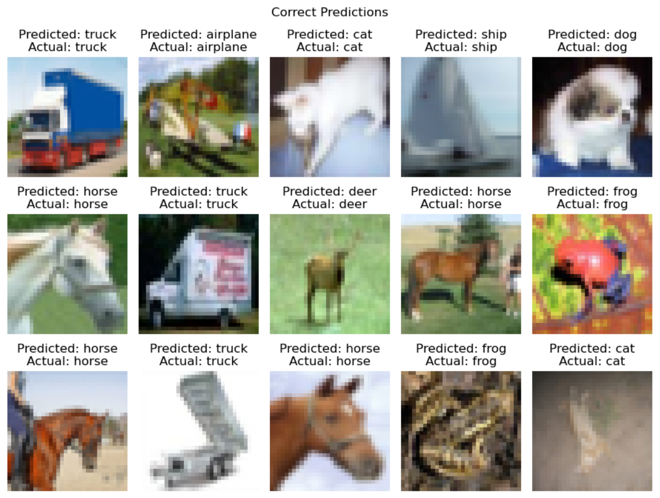

Correct Predictions
Predicted: truck Actual: truck
Predicted: airplane Actual: airplane
Predicted: cat Actual: cat
Predicted: ship Actual: ship
Predicted: dog Actual: dog
Predicted: horse Actual: horse
Predicted: truck Actual: truck
Predicted: deer Actual: deer
Predicted: horse Actual: horse
Predicted: frog Actual: frog
Predicted: horse Actual: horse
Predicted: truck Actual: truck
Predicted: horse Actual: horse
Predicted: frog Actual: frog
Predicted: cat Actual: cat

Fig. 11. Random sample of correct predictions for the baseline transformer model.

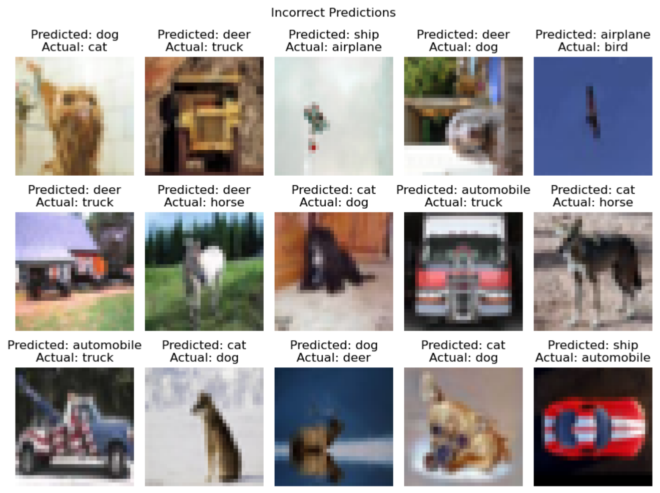

Incorrect Predictions
Predicted: dog Actual: cat
Predicted: deer Actual: truck
Predicted: ship Actual: airplane
Predicted: deer Actual: dog
Predicted: airplane Actual: bird
Predicted: deer Actual: truck
Predicted: deer Actual: horse
Predicted: cat Actual: dog
Predicted: automobile Actual: truck
Predicted: cat Actual: horse
Predicted: automobile Actual: truck
Predicted: cat Actual: dog
Predicted: dog Actual: deer
Predicted: cat Actual: dog
Predicted: ship Actual: automobile

Fig. 12. Random sample of incorrect predictions for the baseline transformer model.

shapes and usually not specific pixels. Regularization was used to prevent overfitting. Decreasing the learning rate was used to try to find a more refined, optimal solution, and stabilize the convergence. Ultimately, the baseline model performed the best. It had the best generalization in terms of accuracy and macro average F1. The other models were trained for less time than the baseline model, but tracking their validation accuracies, they were generally lower than the baseline model at those corresponding points in time. They likely would not outperform the baseline model if given more time, but it is possible. This is something that could not be explored due to the large training times, but the simplicity of the baseline model is also an important factor to weigh in even if the other models had outperformed because it would mean faster predictions and greater ease of explainability.

Regularization performance was weaker than the baseline and adding augmentation always made a dramatic reduction in training performance. Augmented samples likely cause too much distortion since the original images are only 32x32. It would require more time for the training accuracy to be as high as the validation accuracy.

All models struggled with the same classes. For example, cats and dogs were most commonly confused for each other. Still, these transformer models reached test accuracies and F1 scores up to about 96%. This is a significant improvement from ANN and random forest. The drawback would be training time, inference time, and interpretability.

REFERENCES

[1] F. Zhang, ”Once Upon a Time in CIFAR-10,” Medium, Apr. 31, 2022. [Online]. Available: https://franky07724-57962.medium.com/once-upon-a-time-in-cifar-10-c26bb056b4ce

[2] Google, ”vit-base-patch16-224,” Hugging Face, [Online]. Available: https://huggingface.co/google/vit-base-patch16-224. [Accessed: Apr. 14, 2025].