Dl

**Ady**

Good morning, I’m Aditi and we are team 5.

Next Slide

Our team consists of 7 people as shown here.

Next Slide

In this presentation, we are going to talk about the Deep learning and machine models developed to classify images by using the CIFAR-10 data set.

The CIFAR-10 dataset contains 60,000 images in 10 different classes and each class has 6,000 images of each class. For developing a multiclass image classification model for deep learning and machine learning model, this dataset has been used to train and test the data.

Next Slide

For the dl model, the overall approach that we followed was to first load the dataset and split it into training and test sets. We also performed the normalization as a preprocessing part. After that we fine tuned our model by using a pre-trained model and performed the model training and model evaluation.

Next Slide

For both models, data was divided into test sets and training sets. For getting best results training sets have been divided in 80:20 for train and validation data respectively. That was divided to tune the hyperparameters of both models.

Additionally, we have performed normalization to avoid raw data and avoid other problems of the dataset by creating new values and maintaining the general distribution as well as ratio in data. Another reason to implement normalization was to improve performance and accuracy of both models. Here, normalization converts data pixels into a number between 0 and 1. Now, Mingyang will discuss the processes and approach followed in our deep learning model.

**MINGYANG**

Next Slide

We chose ResNet-50 as our pre-trained model. ResNet is a Deep Residual Learning neural network. It improves the efficiency of deep neural networks with more neural layers while minimizing the percentage of errors. The actual ResNet-50 model we imported is pre-trained on the Imagenet dataset by specifying the weights of the resnet50 model. We removed the fully-connected layer at the top of the network for the next step of fine-tuning by setting the include\_top parameters to FALSE.

Next Slide

Speaking of Fine-tuning, we updated the model architecture by removing the previous fully-connected layer heads, providing new, freshly initialized ones, and then training the new Fully Connected layers to predict our input classes.

Next Slide

In the new fully connected layers, we first down-sampled features with Global average pooling 2D. Pooling reduces the size of data, and the number of parameters and improves the ability of the fine-tuned model to detect distorted versions of the object. The next layer is the flattening layer. This layer converts the data into a 1-dimensional array for inputting it to the next layer. Then we add one layer with a rectified linear activation function. It can break the linearity of the input allowing complex relationships in the data to be learned. We also implemented a Dropout layer to reduce the overfitting making model more robust. The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. Finally, we added two more layers with rectified linear and softmax activation functions respectively. Softmax converts a vector of values to a probability distribution. At the end of the fine-tuning process, we compile all the layers into a new fine-tuned model optimised by Gradient descent optimizer. Next Cindy will introduce the evaluation of the test dataset(evaluation of the model).

**CINDY**

Next Slide

During training, we plot the model performance for each epoch. The model is a little overfitting, since our loss value towards validation data doesn’t get any lower than 0.18 after 8 epochs while the loss towards train data keeps decreasing. At the same time, we can see the final accuracy towards validation data remains at 95% even though its accuracy on train data almost reaches 100%.

Next Slide

However, in comparison to the Keras hyperparameter tuning and Vision Transformer we tried before, the current model has already minimized the overfitting. With fine-tuning on ResNet50, we got around a 10% increase in train accuracy and 20% improvement on val acc.

Next Slide

Both datasets look like a good fit in the current model. The model is performing well on unseen data, with 95% test accuracy achieved.

Next Slide

Most confusions arose from ambiguity on features rather than the failure of model architecture. One typical example is about the confusion in distinguishing between cat and dog. 281 ground truths of dogs are predicted as cats. The model is clear about distinguishing between cats with automobiles though, because they have distinct features. Almost no confusion in classifying between automobile and truck.

Such performance tells us the model probably has some unrepresentative training data. We already did cross validation in the fit function. Hence, for future implementation, this can be improved by incorporating data augmentation with random shuffle to increase feature variability in the training data.

Now I will pass to Justin to introduce our machine learning methods.

Next Slide

Part 1: Justin

13: The processes for the ML approach include data loading, preprocessing, feature engineering and splitting, assigning and implementing learning algorithms, comparing accuracy, selecting model, hyperparameter tuning and model evaluation.

14: The first couple of steps are very similar to the DL approach. However, with the dataset is rather complex and the idea of ML is not like DL which are supported by multilayers, we tend to reduce the dimensions and scale the data before injecting them into the appropriate models. As the pixel range of a color image is between 0 and 255, we normalise the data by dividing 255 to make it scale between 0 and 1. And as the data is structured as its row vector equals 32 by 32 by 3 which is 3072, we reshape the arrays from 4 to 2 dimension through using reshape and ravel. And we get each unique image 3072 features. Further, we split the dataset into training, validation and test sets.

Next Slide

**Part 2: KELLY**

**Slide 15**

So what we have done here is that we have performed image classification using four

common machine learning algorithms: Random Forest Classifier, K-Nearest Neighbor (KNN), Decision Tree Classifier, and Naive Bayes Classifier.

Each of the algorithms is implemented with the same process such as applying the same trained and pre-process data to keep the result consistent. All of the algorithms have been evaluated with the test images by obtaining their classification report, confusion matrix and accuracy score.

Next Slide

**Slide 16**

With the evaluation, we gathered all the accuracies of the four ML algorithms and explored them on our dataset, this has been summarized using a bar graph which is shown on the slide.

As you can see, Random Forest Classifier shows the best performance with 47% accuracy

followed by KNN with 34% accuracy, NB with 30% accuracy, and Decision Tree with 27% accuracy. Hence, Random Forest exhibits the best performance and Decision Tree the worst.

My group mate, Chris will take you through the hyper parameter tuning and how it can and has affected the accuracy score of our ML algorithms.

Next Slide

**Part 3: Chris**

**-Slide 17 Random Forest Model Hyperparameter Tuning**

As seen with the comparison of the 4 models, Random Forest had the highest accuracy with 47% and this was our selected model.

To try to improve the accuracy of the selected model we performed hyperparameter tuning, by adjusting the n\_estimators, which is the number of Decision Trees in the Random Forest.

Using GridSearchCV, the Random Forest model was tuned with the validation data, and parameter values set at 150 and 200.

With 3 cv folds, this was a total of 6 fits for the GridSearch.

These parameters were selected using Random Search.

As seen in the sample code. The best parameter Grid Search result was n\_estimators at 200.

The updated parameter was applied to the Random Forest model and the results had an improved accuracy of 48%.

Pass it on to talk about the evaluation of results and the conclusion

Next Slide

**Part 4: jiangli**

**-Slide 18 Random Forest Model Evaluation**

With hyperparameters tuned, the Random Forest model had an improved accuracy of 48%. From the confusion matrix, it can be observed that this model had better accuracy predicting aeroplanes, automobiles, ships and trucks compared to cats, birds, deers and dogs.

Next Slide

**-Slide 19 Test Case**

Here we have a sample of the test case results. With an accuracy of 48%, there is a high probability that the Random Forest model will have an incorrect prediction. We can see that in this sample as there are some variations in what is predicted correctly, with some trucks, ships and frogs predicted correctly, and some dogs, cats and birds predicted incorrectly.

Next Slide

**-Slide 20 Main Findings**

After comparing the results of the Deep Learning and Machine Learning models, we can conclude that the Deep Learning CNN model is much more suitable for multi-class image classification compared to the Machine Learning Random Forest model.

With the test data set, Deep Learning has an accuracy of 95% while Machine Learning has an accuracy of 48%.

The CNN model requires more time to fine tune but it has much better performance with larger and complex data sets compared to Machine Learning models.

We would recommend using Deep Learning CNN ResNet50 for multi-class image classification on large image sets such as CIFAR-10.

Next Slide

**-Slide 21 Test Results**

Here are the accuracy and confusion matrix results of our models.

Next Slide

Thank you!