CS412 Term Project
Spring 2023

Can Ceylan 29010

Barış Ulaş Çukur 29461

Alper Kaan Odabaşoğlu 28147

Safa Abdullah Söğütlügil 29214

Pelinsu Saraç 28820

kNN Notebook Link:

https://drive.google.com/file/d/1xRLETk1ns6NAQl6Dhcv7fIaSzUtwOCjx/view?usp=sharing

Decision Tree link:

https://drive.google.com/file/d/1qDzmcy86M5gc9LByi-N7_jD8UMn1da_V/view?usp=sharing

CNN Notebook link:

https://drive.google.com/file/d/1CJCmLiQlIWPXmRh4RNMmO8xOHV7yNsGF/view?usp=shar
ing

1. Abstract

kNN, decision tree and CNN machine learning models were constructed to challenge the image classification problem using the CIFAR10 dataset, which consists of 60.000 images, belonging to 10 different classes.

Among the machine learning models, the best model turned out to be CNN with 0.8737 accuracy. The main reason CNN was able to perform much better on this task was its ability to consider different pixels(data points) together, whereas such pixels are only evaluated individually with kNN and decision trees.

The study also highlights the weakness of kNN and decision tree models to be used on image classification tasks. The kNN model falls short since it does not learn the features of the dataset but rather simply calculate the distance between each datapoint. As for the decision tree model, it underperforms relative to CNN since it considers pixels individually and misses the special features of the images, which is crucial for the image classification task.

2. Introduction

Main objective of this project is to create image classifiers with a popular machine learning dataset, CIFAR-10. CIFAR-10 is a benchmark image dataset for machine learning tasks. It includes 60,000 colored images which belong to one of the 10 classes. 60,000 images split into two sets: 50,000 for training and 10,000 for testing. Further information about this dataset can be found in the next section.

The classification task for machine learning, in most simple words, is to predict the class/category of a sample from a dataset, given the features that the sample has. For an image dataset like CIFAR-10, this task is basically finding the category of the object shown in the image. Image classification is an important task because it has many possible daily-life applications. For example, image classification has already been used in crucial fields such as medical diagnosis, security, and surveillance.

To achieve this objective, different machine learning methods were tried. These are k-nearest-neighbor (KNN), decision tree, and convolutional neural networks (CNN). At the end of the report, the best model among all tried algorithms is also discussed based on their performances.

3. Dataset

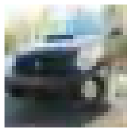
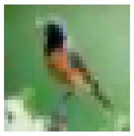
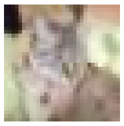
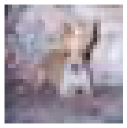
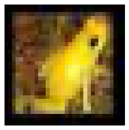
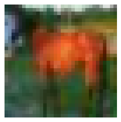
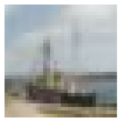
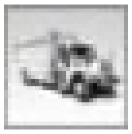
We used the CIFAR-10 dataset which consists of 60.000 32x32 colored images in 10 classes. You can see an example for each class in the table below (Note that they are enlarged images as the original size is too tiny). The dataset is split into 6 batches with 10.000 images each. Five of these batches are training batches whereas one of them is for testing. Hence, training and test batches have 50.000 and 10.000 images respectively.

The test batch is balanced, i.e. it has 1000 images from each class. In total, the training batches have 5000 images from each class but they are not evenly distributed among batches. So, some classes are more prevalent than others in certain batches. Also, the dataset is mutually exclusive, i.e. there are no images that belong to more than one class.

The data is a numpy array of 10.000x3072 of 8-bit unsigned integers with each row in the array containing a 32x32 colored image.

The color information is divided into three sections within each row: the first 1024 values are of the red channel, the next 1024 green channel, and the final 1024 of the blue channel. The images are stored in row-major order. This means that the first 32 entries in the array represent the red channel values of the first row of the image.

The labels list contains 10.000 numbers ranging from 0 to 9. As the name suggests, these are the label information of the aforementioned data list. These are in the same order as the data list and their meaningful names such as automobile, cat, etc. are held in the batches.meta file that has the label_names list with a total of 10 elements (see below).

Airplane	**Automobile**	**Bird**	**Cat**	**Deer**
				
Dog	**Frog**	**Horse**	**Ship**	**Truck**
				

4. Methodology

kNN, Decision Tree, and CNN machine learning algorithms were explored.

kNN:

Dataset was compressed, so it needed to be extracted with the !tar operator. Data was stored in a dictionary format with 4 keys, but only 2 keys were relevant so only those were copied into 6 dictionary variables. The value of k needs to be tuned since it affects the models performance directly. During hyper-parameter selection,which involved training 6 models with different k values (1,3,5,7,9,11) for each fold, 5 folds were used. Since the data already came in 6 splits, each containing 10000 elements, splitting any training set was not necessary. During the hyper-parameter selection, the 4 folds had to be combined to form the training test, and the unused fold would be used as the validation set. After the hyper-parameter was chosen, the 5 splits were combined to train the model, and the model's accuracy was calculated with the unseen 6th set.

K = 7 was chosen as it had the best validation accuracy.

Final model accuracy: 0.2962

Decision Tree:

Another model that was tried is the Decision Tree. In short, Decision Trees split the training data by asking questions about the features of the data in order to classify. Splits are done to minimize the impurity. Thus, first hyperparameter chosen for fine-tuning is the impurity criterion, which can be *gini index* or *entropy*. Second hyperparameter chosen is the maximum depth of the tree. It is chosen as it is known that if the tree is not stopped at some point, it can grow very deep and overfit to data. Before the fine-tuning, an initial model was trained and it was observed that it has depth of 44. Thus, for the tuning, values of 10, 20 and 30 were used. Next to them, *None* was also added, to see whether it is indeed a good idea to set maximum depth. One remark is that, no preprocessing on data is applied for this algorithm.

To conduct the fine-tuning in a more efficient way, grid search object provided by Sklearn library was utilized. With the help of this object, 5-fold cross validation was done. Best hyperparameter pair turned out to be *gini index as criterion* and *maximum tree depth of 10*. One final model is trained with these hyperparameters and the final accuracy on the test set is 0.30. This value can be considered as lower than expected, thus it can be said that the decision tree does not seem to be suitable for image classification.

CNN:

Another ML model that was tried was CNN, which yielded the best accuracy among all the other models. To start, data labels were needed to be applied to the dataset. The dataset was iterated for each batch and the labels were stored in two arrays, namely *train_images_batch* and *train_labels_batch.* Label names were also stored in the *label_names* array. After the iteration, *np.concatenate* was used to combine the image and label arrays into the final training dataset. To prepare the images for training with CNN, the shape of *train_images* is reshaped and transformed.

After reshaping the dataset, a random image was chosen (with index 123) to determine the success of the operations. This was the result:

After reshaping the dataset, it was normalized by dividing each element of the train and test list by *255.0*. The *val_images* and *train_images* lists were then created with the last 10.000 and the first 40.000 elements of the *train_images* list respectively. Similarly, the labels were also split into two groups consisting of the last 10.000 and the first 40.000 image labels created from the previously formed *train_labels* list.

The dataset was then encoded using *one hot encoding* method, which tensorflow provides with *to_categorical* function. The reason why we use one-hot-encoding is to improve the performance of the model as much as possible. Categorical data is better understood when it was encoded in this way.

Later, the data augmentation tasks were performed to reduce overfitting using the *ImageDataGenerator* function of the tensorflow library. The parameters used in ImageDataGenerator function width_shift_range, height_shift_range and horizontal flip. Width_shift_range determines the random horizontal shift of the image. It is given as 0.1 to the generator which ensures the image can be shifted horizontally by a maximum of 10%. Height shift range is the vertical version of the width_shift_range. As the name suggests, horizontal flip handles the horizontal flipping action.

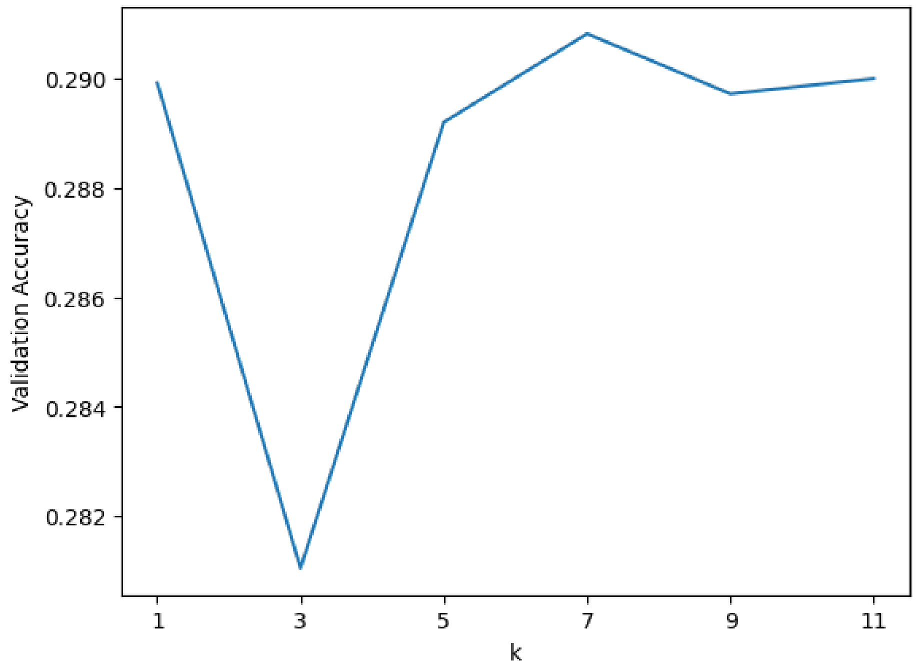
During the training phase, the convolutional layers are artificially generated. From keras.models, Sequential is imported to initialize the model. From keras.layers, Conv2D, MaxPooling2D, Flatten, BatchNormalization functions are imported which are used for defining and constructing the convolutional layer structure.Parameter padding=’same’ is used for the purpose of the elimination of dimension differences. Moreover, ‘he-uniform’ kernel_initializer is used to better fit for ReLu activation function. In the convolutional layers and initial part of dense layers, ReLu activation function is used. However in the last dense layer which is the output layer where 10 classes are specified for the model, softmax activation function is used. The reason why it is utilized for the output layer is the high performance of softmax on categorical output layers. Furthermore, it is better in terms of probability normalization. As an optimizer, Stochastic Gradient Descent method is preferred with 0.001 learning rate. In addition, momentum is preferred as a parameter to overcome limitations of the optimizer. It is initialized as 0.9 where 90% of the previous updates transferred to the current update. CNN model is compiled with the SGD optimizer, categorical-crossentropy loss function and ‘accuracy’ metric. During fitting, augmented train and validation data is given to the model. Batch size is determined as 64 which accelerates the training phase by locating more data into bigger batches. Epoch count is given as 500 as the model was trained from the scratch. Smaller epoch counts led to lower accuracy acquisition. Last of all, steps per epoch is used to accelerate the training phase more.

Pickle library was also utilized to save the trained model locally, which enabled the session to not to train the model all over again after it was closed.

5. Experiments

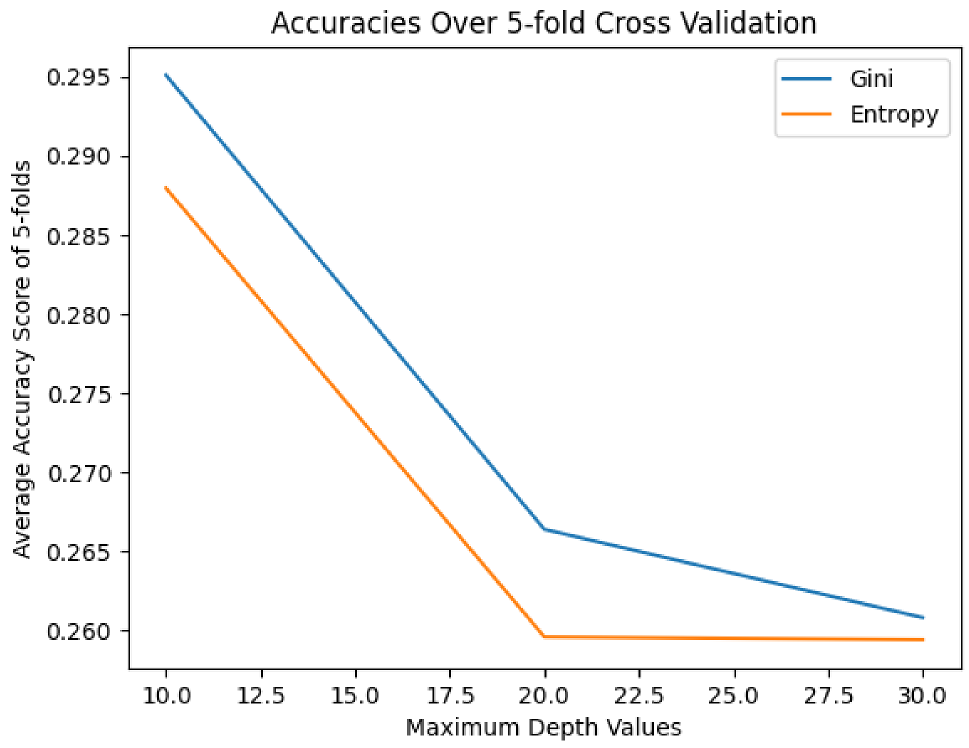
kNN:

K Value	**Mean Validation Accuracy over 5-folds**
1	0.28992
3	0.28104
5	0.2892
7	0.2908
9	0.2897
11	0.29



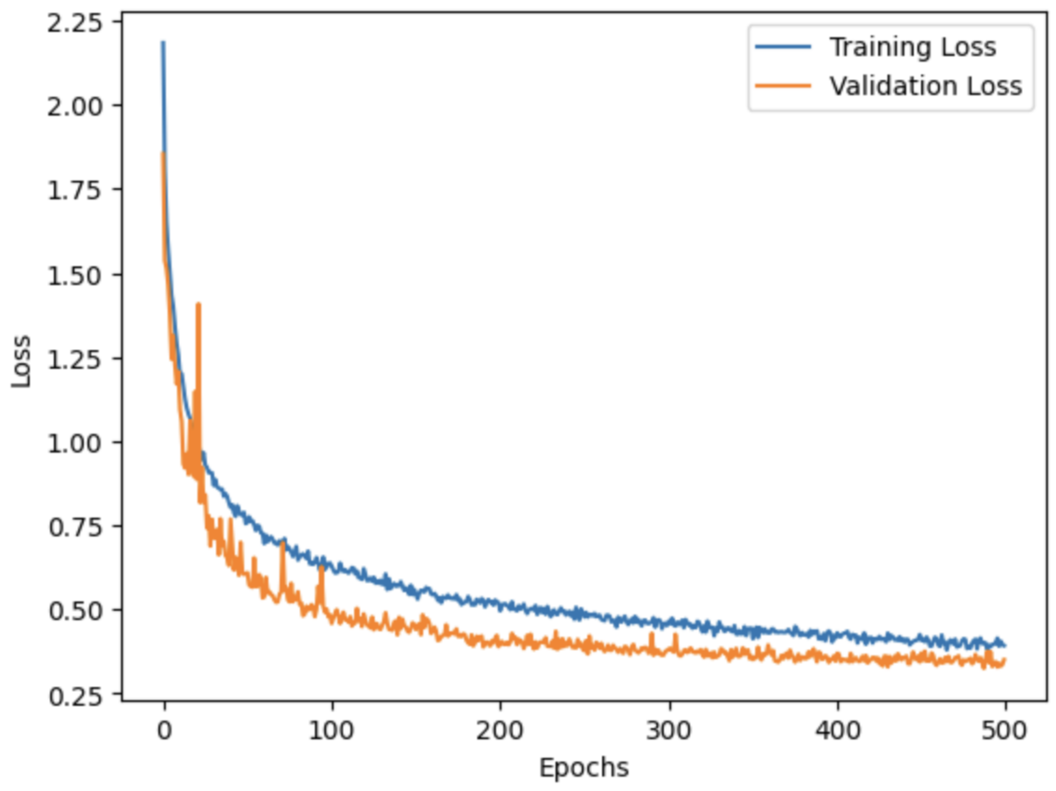
Decision Tree:

Rank	Hyperparameter Pair	Mean Acc. over 5-folds
1	gini-10	0.29508
2	entropy-10	0.28794
3	gini-20	0.26638
4	gini-30	0.26080
5	gini-None	0.25996
6	entropy-None	0.25972
7	entropy-20	0.25958
8	entropy-30	0.25940



CNN:

Epoch	Validation Accuracy
1	0.3420
20	0.6142
50	0.7903
100	0.8364
200	0.8669
300	0.8746
400	0.8835
500	0.8817



Model Summary:

Model	Final Accuracy
kNN	0.2962
Decision Tree	0.3061
CNN	0.8737

6. Discussion

The CNN model outperformed kNN and decision tree models in image classification. For the CIFAR-10 dataset, the CNN model's advantage can be reasoned by its ability to analyze complex features better than other models, especially ones spread over multiple pixels. This is achieved by convolutional layers of the CNN. Other models(kNN, decision tree) do not have the ability to examine the relationship between different pairs of data points. It can also be stated that CNN model is better in terms of handling data with higher dimensions, again due to the evaluation of pixels together rather than individually. It must also be noted that, during the training of the CNN model, if more epochs were used, for example up to 1000, the model can have better validation accuracy results. Further studies can be conducted with higher computational power to achieve this aim.

7. Conclusion

The kNN algorithm had 0.2962 accuracy, and since it is greater than 1/10, we can state that the model actually learned something, although it can be considered to be a weak learner compared to the CNN algorithm which had an accuracy of 0.8737. The kNN model is concluded to be significantly weak compared to CNN, and the main reason that causes such a massive gap between these 2 models is the fact that kNN only calculates the distance between each data point without actually trying to understand any features of the dataset. This simplicity of the kNN machine learning model may still provide favorable results in some problems, but in this image classification problem, it turns out that it is a major weakness of the kNN machine learning model.

Another model that was tested was Decision Tree. Similar to kNN, it also did not have great results compared to CNN. Final test accuracy obtained with the decision tree algorithm was 0.3061. One possible reason for why the Decision Tree model did not perform well for the image classification task is related to how the algorithm evaluates the pixels. For the decision tree, each pixel value of the flattened image is an independent feature. Thus, each pixel was evaluated individually in a sense. However, thinking from the CNN perspective, a true feature distinguishing the image can be spread over multiple pixels and CNN is able to find them whereas Decision Tree cannot. This can be summed up as lack of spatial information for the decision tree. In the end, it can be said that Decision Tree is not a suitable algorithm most of the time. Other reasons can be, the decision tree is affected by the dataset heavily. For example, even if two images show the same object, due to the lightning of the images they could be evaluated differently by the model, as pixel values would be slightly different. Finally, it is known that the Decision Tree algorithm is prone to overfitting. With this many features (3072 pixels), it is quite possible for the model to overfit easily.

In conclusion, CNN model outperformed other two models, kNN and Decision Tree. This was an expected result, due to the reasons discussed in the Discussion section above. However, the accuracy of the kNN and Decision Tree model hovered around 30%, while CNN had about 87% accuracy. This much difference was not expected, which goes to show how much more advanced CNN machine learning model is compared to models such as kNN and Decision Tree.