European Traffic Signs Images Classification Using Convolutional Neural Network

INTRODUCTION

The problem given was related to training a model that can classify images of European road traffic signs. The image data given for this task was a modified version of the Belgium Traffic Sign Classification Benchmark. The modified dataset contained grey-scale images of size 28x28 and requirement of the task was to train a model which is capable of classifying images according to sign-shape, such as diamond, hex, rectangle, round, triangle and another model which is capable of classifying images according to sign-type, such as stop, speed, warning, parking, etc. The task is formulated as a multi-class supervised classification problem.

METHODOLOGY

The parent directory and all the subdirectories were uploaded into the Jupiter Notebook which was running on AWS SageMaker service. Using a loop, all the image was loaded into a data frame with their labels (Both shape and corresponding sign). There were in total 3699 images including all the subdirectories with 5 shape labels and 16 sign labels.

Task 1: Classify images according to sign-shape

On further analysis, it was found that the dataset was dominated by round-shaped images with 1760 samples and hex-shaped samples were too low relatively. The exact numbers can be referred from Figure 1. Based on the samples, we address an issue that the label classes were imbalanced as there were only 43 samples for hex-shaped signs. This issue could be fixed using the image augmentation techniques. Image augmentation artificially creates variations in existing images to expand the dataset. However, this was only documented in the report and not performed because of the time constraints.

Task 2: Classify images according to sign-type

Using the same dataset images were loaded into another Jupiter Notebook assigned for this task. Here we had 16 unique labels or sign types that our model had to learn from and classify them in the later stage. From Figure 8, it can be observed warning sign samples were in majority with 695 instances, and stop sign has the lowest with 43 samples. Still, the class labels were found imbalanced but not as much as we noticed for the shape labels.

As we used a similar approach for both the task, therefore, a further report will explain the general approach applied to both the task. However, the section will be subdivided if there's a significant difference between the approach.

Train, Test and Validation Split

Moving forward, the dataset was divided into 3 subsets i.e. train, validation, and test. To perform this task, hold-out cross-validation was used to first divide the dataset into 80(Train):20(Test) ratio, and the train was further divided into 80(Train):20(Validation) ratio.

Images in all the subset were rescaled and normalized using ImageDataGenerator from the TensorFlow library. Usually, each pixel in an image varies from 0-255 but it's highly recommended to normalize them in the range of 0-1 to speed up the model training process. In addition to this, the batch size was kept at 32. Batch size is a hyperparameter of Stochastic Gradient Descent, using which you can define the number of samples to work through before updating the internal model parameters. In our case, the data will load from data generators into batches of size 32, and by the end of each batch, prediction of the model will be compared with expected output variables and based on this, the error rate will be calculated. This is one of the famous techniques when working with a large dataset for efficiently using the system's memory. The data generators were defined for the train, validation, and test dataset.

MODEDLING

To solve the problem of multi-class supervised classification problem, a different model was trained using different Layers, Complexity and other parameters for predictions. CNN was chosen over MLP was CNN is known to perform better on complicated image classification problem and is much more flexible in terms of setting the parameters in the architecture. However, CNN includes lots of different types of hyperparameter tuning. Following section will discuss how the various types of decisions were made either using experiments or literature.

Number of Layer and Nodes

There's a theoretical finding by Lippmann in the 1987 paper "An introduction to computing with neural nets" [1] which demonstrates that an MLP with two hidden layers is enough for creating classification regions on any desired shape. Our first task was to classify the shapes by creating classification regions around them according to their shapes and therefore, such findings were taken into account while modeling. However, few more complex models (with 3 hidden layers) were trained with hyperparameter tuning as an experiment to challenge the finding by Lippmann.

Activation Function

Activation function "relu" (Rectified linear unit) was applied between the dense layer, which is the most used non-linear activation function in neural networks. We used relu for the hidden layers over other activation functions like tanh and sigmoid as relu is much faster and hence, accelerates the convergence of SGD. In addition to that, relu doesn't have vanishing gradient problems like tanh or Sigmoid function.[4]

Also, "SoftMax" function was added in the end as it is more suitable for the output layer and returns the output in between 0 and 1 which is really helpful in model that has to predict the probability of class labels.

Pooling Function

There was a need for a function to progressively reduce the spatial size of the representation so that the number of parameters and computation in the network can be reduced. This would make the model training faster while consuming less memory. Therefore, we used "MaxPooling" in between the hidden layers which is the most popular pooling method in neural network modeling. MaxPooling works by choosing the maximum grid value in order to extracts the most important features. The other know pooling method is "Average Pooling".

Average pooling takes all the grid values into account and flows the mean value to the next layer which results in a generalized computation. In other words, it takes all the pixels of an image into account while training the model, unlike MaxPooling method. In our problem, we wanted to consider only important features like edges and not the pixels around the traffic signs. Therefore, for this problem, MaxPooling seemed a better option to gain better results.

Loss Function

We selected "categorical_crossentropy" as our loss function to deal with the multi-class classification problem. It is a default loss function and was chosen over other loss functions like "sparse_categorical_crossentropy" loss function as we had only 5 class labels in this task which can be easily encoded numerically without taking significant memory. However, "sparse_categorical_crossentropy" could be more suitable for task 2 as there were 16 class labels.[3]

Optimizer

Two types of optimizers were tried out using an experimental approach and the model was selected based on higher accuracy and their convergence. The first one was "SGD" (Stochastic Gradient Descent) optimizer to find parameters that reduce the loss while optimizing the model in Neural Networks by taking the average gradient on all pairs of samples in the training set. Another one was Adam (Adaptive Moment Estimation) optimizer, which was used to make model computationally efficient and deal with problems of noisy and sparse gradients in an efficient manner. Adam is known for its ability to find a global minimum point. Also, over SGD, in our case, it was expected to Adam to give better performance because of its robustness to the noisy data. The given dataset for this task had many noisy images as shown in Figure 2.

The learning rate parameter of both optimizers was set to 0.01 and the momentum of 0.9 to help gradient vectors accelerate in the right direction, thus leading to faster converging.

Regularization

The possibility of over-fitting in our model was taken into consideration. A model can be considered as over-fitted if your model performs well on training data (seen data) but poorly when applied to unseen data. In other words, this is a case of high bias where a model starts to memorize the training data instead of learning the relationship between features and labels. To handle such a situation, we used techniques likes L1 regularization, L2 regularization, and dropout regularization. [5]

The dropout layer was set to 0.5, which means it will set 50% of the features as zero while training, however, during validation all the features will be used which could result in more robust and high accuracy. That's the reason why validation accuracy came higher at the early stages of epochs for almost every time we used the dropout layer in or model. It helped our model to be less sensitive against the specific weights of neurons which in return provided a model that was capable of generalizing the images well and less likely to overfit the training data.

Epochs and Batch Size

One epoch is when an entire data is sent forward and backward through the neural network only once and this epoch could be too big to feed into the memory, therefore, we use batch size to divide the data into several batches [2].

However, it a crucial decision in the process to decide the number of epochs for training the model to a satisfactory point. To diagnose this, we have used learning curves to know whether the model has over-learned, under learned, or is suitably fit to the training dataset. An ideal epoch size would be a point where the training and validation accuracy start to plateau as they converge. Lesser the epoch, lesser will be the computational time. Based on the diagnosis of the learning curve graph, we found an ideal epoch to be around 30 while doing task 1. However, it was increased to 50 to make sure the stability of the convergence. For task 2, the epoch was increased to 100 as the learning curve was dropping (case of overfitting) for few models at the end of epoch 50. Therefore, we tuned those models and increased the epoch to 100 to see the impact of the tuning on the long run of the model.

In addition to this, larger batch size results in faster progress in training, but end up converging slower. On the other hand, smaller batch sizes train slower but capable of converging faster. Therefore, we kept our batch size to 32 considering the size of our dataset and for faster convergence so that epoch could be decreased to a lower number.

Performance Metrics

As mentioned earlier, the dataset had an issue of imbalanced class labels. Due to this issue, we didn't use "Accuracy", the most popular metric for classification problems, as this metric can be misguiding in the case of imbalanced class labels. Instead, we used F1-score as our metric to measure and compare the performance for different trained models when predicting the unseen data. F1-score is known to work better for imbalanced classification scenario as it is the harmonic average of the precision and recall.

EVALUATION

Task 1 Evaluation: Classify images according to sign-shape

Almost every trained CNN model performed well and gave an excellent performance on unseen data (testing) which can be seen by referring to Table 1. However, the best model was selected based on the observation made on their learning curves and F1-score and most importantly, their simplicity following the principle of parsimony.

The first two models which were trained with the convolution layer were named model_cnn_2L and model_cnn_3L. Details of used hyperparameters can be seen in Table 1. They both produced the weighted F1-score of 99% on the prediction with testing data. However, when we looked at the learning curve of both the models, we observed that for model_cnn_3L, validation accuracy was passing over the training accuracy during the starting of epoch and then later converges with the training learning curve (Figure 4). The same effects were seen in the model_cnn_2L, shown in Figure 5, but the validation learning curve was much closer to the training curve when compared. Also, when we analyzed the losses from the trained model, model_cnn_2L gave the least loss with a score of 0.0170

We further tuned the model by first dropping the dropout layer to analyze the difference. The performance of the tuned model was also recorded in Table 1. The validation accuracy decreased on the removal of the dropout layer which showed that our model was prone to over-fitting and hence, some type of regularization was needed in our model. As we discussed in the earlier section that the model will be trained using both SGD and Adam optimizer, therefore another model was trained "model_cnn_2L_v3" with the same configuration but with Adam optimizer. This model resulted in better performance in terms of accuracy, losses, F1-score stability with the shuffled dataset, and learning curve convergence. In order to challenge the

Lippmann's finding, we trained more complex and denser models with 3 layers and even applied different regularization techniques to improve their accuracy. However, none of them performed better than our selected 2-layered model.

Hence, we failed to prove the Lippmann's finding wrong as the neural network with 2 hidden layers worked best for us, and therefore, model_cnn_2L_v3 was selected to be used for our independent evaluation in the later stage to classify the shape of the traffic signs.

Task 2 Evaluation: Classify images according to sign-type

The results from the different trained models are stored in Table 2 which can be referred from the Appendix section of the report. Model model_cnn_2L_adam performed the best with 0.9934 accuracy on validation dataset and 0.031 losses which was the least score among all other model. It was also observed that the accuracy was dropping in the denser layered CNN models (example shown in Figure 7) and also, it was necessary to apply any type of regularization else the model was overfitting. In an experiment to increase the accuracy, Average Pooling was also tried in model_cnn_2L_v4 in a hope to use all the pixel values of the dataset. It performed well but not better than our selected model. Our selected model was showed stability when training and validation learning curves were compared with all the over models. Therefore, we again made our choice based on simplicity and performance.

ULTIMATE JUDGEMENT

Task 1 Judgement

Based on all the observations and experiments, model_cnn_2L_v3 came out to be a model to classify the shapes of the traffic signs and I recommend model_cnn_2L_v3 model to solve this particular problem. This judgement was made on the following criteria:

- High accuracy and low losses in validation data with less complexity.
- No signs of over-fitting or dropping accuracy.
- Able to generalize well to the unseen data with 99% F1-score which can be even improved if we increase the samples for hex-shaped class labels.
- Stable performance, irrespective of data shuffling which showed that the model is actually learning the relationship between the features and labels.

Drawback:

- The number of trainable parameters was high (61,157) i.e. almost double if we use a denser model, and therefore, computational requirements are bit higher than other models.

Task 2 Judgement

Based on all the observations and experiments, model_cnn_2L_adam performed better than other trained models, and therefore, I recommend this model. This judgement was made on the basis of model's performance, stability and the fact that it was less prone to overfit.

Drawback:

- As the number of epochs increased to 100 for this task, therefore computational time would be a bit higher in comparison to the model trained in the previous task.

- CNN is known to detect edges better than components insides the objects. Therefore, this model may not perform well in real-world settings and should be only considered for the original dataset used for this particular task.

INDEPENDENT EVALUATION

To perform this evaluation, images from different sources were taken and loaded into the notebook. The images were gray-scaled and resized our target size i.e. 28x28 before putting them into the image generator. Using the image generator, the images were normalized and then feed into the prediction function of our selected model. There were 130+ samples and the majority of them belonged to round-shaped signs as round shaped images were easily available on the internet in comparison to other types of images. For a few shapes and signs, there were only a handful of images because of their limited presence on the internet.

Task 1 Independent Evaluation

When our selected model (model_cnn_2L_v3) was applied on a set of independently collected data, we got an accuracy of 0.87 and F1-score of 0.86. This showed that our model performed reasonably good and not very accurate to be in a real-world setting despite performing exceptionally on validation and testing dataset. It was felt that our model needs to be trained on more samples in order to generalize the images from the real world.

Task 2 Independent Evaluation

When our selected model (model_cnn_2L_adam) was applied to the independent dataset, we got an accuracy of 0.69 and F1-score of 0.61, shown in Figure 9. This showed that our model wasn't fit for the real-world setting was it performed poorly. Further analysis and research were done in order to understand the reason behind this.

When the confusion matrix was checked to identify the issues with the performance, it was identified that the round class gave 0.91 F1-score predictions and hex gave 0.59, shown in Figure 6. Therefore, it was clear that one of the reasons for this problem caused due to the imbalance present in the class labels of the trained dataset because samples with round shapes are in majority and definitely there was a need for more samples for other class labels. Image augmentation is one of the approaches which could be used to increase the sample size for these images using the existing data. However, that will increase the samples of round shaped images also. To tackle this situation, the weight balancing technique could be used to place more weights only on minority classes.

On further investigation of the lack of accuracy, it was realized that CNN has few limitations such as CNN doesn't encode the position and orientation of objects. In addition to this, CNN finds difficulty to learn from images when the same image is taken from different angles as it doesn't have a built-in understanding of 3D space. A human brain needs to see an object dozen of times in order to detect an object from different orientations, however, CNN needs tens of thousands of examples to produce a good performance [6]. In this case, 3699 images are much lesser to perform in real-world settings. Maybe me more complex model (with 4-5 layers) could have helped the model to learn and generalize the details inside the images well, to achieve a little increase in the accuracy. But, was it really worth to achieve that accuracy with a significant increase in the computation cost? Well, this is totally subjective and based on real-world setting in terms of money and provided computational power.

APPENDICES

Epoch: 50 Batch Size: 32	Training & Validation			Hyperparameters						Predictions
Model Name	Trainable parameters	Train Accuracy: Loss	Validation Accuracy: Loss	Hidden Layers	Activation function	Pooling	Loss function	Optimizer	Weight optimization	Weighted F1-scoore
MLP	54,725	0.9822: 0.0851	0.9473: 0.1600	2	Relu	-	categorical_ crossentropy	Adam	-	0.94
model_cnn_2L	61,157	0.9945 :01453	0.9917 :0170	2	Relu	Max	categorical_ crossentropy	SGD	Dropout(0.5)	0.99
Model_cnn_2L_v2 (tuned)	61,157	0.9997: 0.0005	0.9868: 0.0230	2	Relu	Max	categorical_ crossentropy	SGD	-	0.99
Model_cnn_2L_v3 (tuned)	61,157	0.9957: 0.0117	0.9934: 0.0177	2	Relu	Max	categorical_ crossentropy	Adam	Dropout(0.5)	0.99
model_cnn_3L	32,549	0.9956: 0.1493	0.9917: 0.0369	3	Relu	Max	categorical_ crossentropy	SGD	Dropout(0.5)	0.99
model_cnn_3L_v2 (tuned)	32,549	0.9991: 0.2940	0.9950: 0.5555	3	Relu	Max	categorical_ crossentropy	Adam	L2, Dropout	0.98
model_cnn_l1_reg	32,549	0.9983: 0.0205	0.9934: 0.0369	3	Relu	Max	categorical_ crossentropy	SGD	L1 & Dropout	0.99

Table 1: Results from trained model for task 1

```
records with diamond shape is  282
records with hex shape is  43
records with round shape is  1760
records with square shape is  688
records with triange shape is  926
```

Figure 1: Sample records for each shape

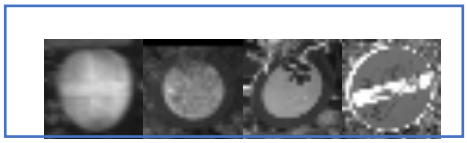

Figure 2: Noisy Image Data

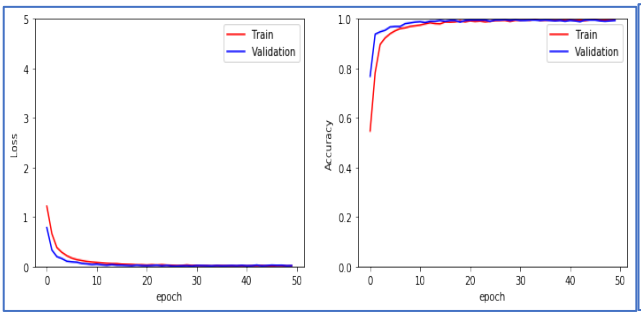

Figure 5: Model_cnn_2L_v3 learning curves

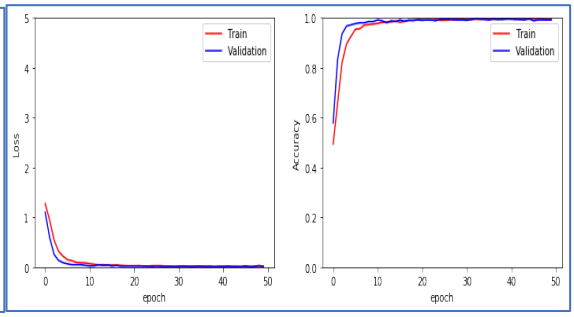

Figure 4: Model_cnn_2L learning curves

```
Prediction shape is (136, 5)
              precision    recall  f1-score   support

     diamond       0.57      0.67      0.62         6
         hex       1.00      0.42      0.59        12
       round       0.91      0.92      0.91        99
      square       0.69      0.85      0.76        13
    triangle       0.75      1.00      0.86         6

   micro avg       0.86      0.86      0.86       136
   macro avg       0.78      0.77      0.75       136
weighted avg       0.87      0.86      0.86       136

[[ 4  0  2  0  0]
 [ 0  5  7  0  0]
 [ 3  0 91  5  0]
 [ 0  0  0 11  2]
 [ 0  0  0  0  6]]
```

Figure 6: Task 1 independent confusion Matrix

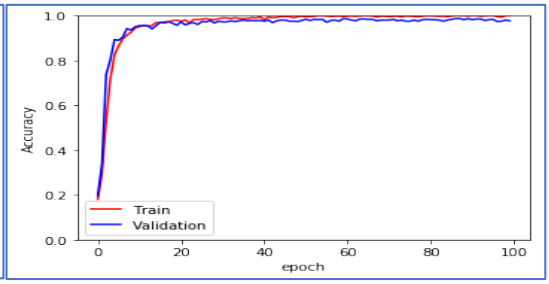

Figure 7: Model_cnn_3L accuracy curve

Epoch: 100 Batch Size: 32	Training & Validation			Hyperparameters						Predictions
Model Name	Trainable parameters	Train Accuracy: Loss	Validation Accuracy: Loss	Hidden Layers	Activation function	Pooling	Loss function	Optimizer	Weight optimization	Weighted F1-scoore
Model_cnn_2L	61,872	0.9866: 0.0372	0.9884: 0.0437	2	Relu	Max	categorical_ crossentropy	SGD	-	0.98
Model_cnn_2L_v2 (tuned)	61,872	1.0: 0.0001	0.9736: 0.1865	2	Relu	Max	categorical_ crossentropy	SGD	Dropout(0.3)	0.98
Model_cnn_2L_ adam (tuned)	61,872	0.9987: 0.0032	0.9934: 0.0317	2	Relu	Max	categorical_ crossentropy	Adam	Dropout(0.5)	0.98
Model_cnn_2L_v4	61,872	0.9995: 0.0017	0.9856: 0.0415	2	Relu	Average	categorical_ crossentropy	RMSprop	Dropout(0.5)	0.99
model_cnn_3L	38,448	1.0: 0.0024	0.9769: 0.0767	3	Relu	Max	categorical_ crossentropy	SGD	Dropout(0.5)	0.99
model_cnn_3L_v2 (tuned)	21,968	0.9940: 0.0489	0.9654: 0.1587	3	Relu	Max	categorical_ crossentropy	SGD	L2, Dropout	0.97

Table 2: Results from trained model for task 2

```
records with warning sign is  695
records with roundabout sign is  98
records with trafficdirective sign is  195
records with noentry sign is  375
records with parking sign is  276
records with bicycle sign is  285
records with laneend sign is  118
records with giveway sign is  231
records with crossing sign is  95
records with continue sign is  199
records with speed sign is  316
records with noparking sign is  242
records with rightofway sign is  282
records with limitedtraffic sign is  125
records with stop sign is  43
records with traveldirection sign is  124
```

Figure 8: Sample records for each sign

```
Prediction shape is (136, 16)
                precision    recall  f1-score   support

       bicycle       0.47      0.70      0.56        10
      continue       0.33      0.50      0.40         2
      crossing       0.00      0.00      0.00         6
       giveway       1.00      1.00      1.00         1
       laneend       1.00      1.00      1.00         1
limitedtraffic       1.00      1.00      1.00         3
       noentry       0.63      0.92      0.75        13
     noparking       0.46      0.86      0.60         7
       parking       0.75      0.75      0.75         4
     rightofway       0.36      0.67      0.47         6
     roundabout       0.86      0.40      0.55        15
          speed       0.90      0.95      0.93        20
           stop       1.00      0.50      0.67        12
trafficdirective       0.82      0.56      0.67        16
 traveldirection       0.60      0.20      0.30        15
        warning       0.31      1.00      0.48         5

      micro avg       0.63      0.63      0.63       136
      macro avg       0.66      0.69      0.63       136
   weighted avg       0.69      0.63      0.61       136
```

Figure 9: Task 2 independent confusion Matrix

References

[1] Brownlee, J. (2020). How to Configure the Number of Layers and Nodes in a Neural Network. Retrieved 28 May 2020, from https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

[2] Brownlee, J. (2020). Difference Between a Batch and an Epoch in a Neural Network. Retrieved 28 May 2020, from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

[3] Brownlee, J. (2020). How to Choose Loss Functions When Training Deep Learning Neural Networks. Retrieved 28 May 2020, from https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[4] Complete Guide of Activation Functions. (2020). Retrieved 28 May 2020, from https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

[5] Preventing Deep Neural Network from Overfitting. (2020). Retrieved 28 May 2020, from https://towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a

[6] Sumit, S. (2020). Drawbacks of Convolutional Neural Networks. Retrieved 28 May 2020, from https://sakhawathsumit.github.io/sumit.log/2018/07/21/drawbacks-of-convolutional-neural-networks.html