DEEP LEARNING - 96943

INSTRUCTOR: DR. TAHIR SYED

Cat vs Dog vs Bird Kaggle Competition Assignment # 3

Authors:
Abdul Haseeb (ID: 17132)
Annayah Usman (ID: 18673)

Date: December 16, 2024

Contents

1 Introduction

This project focuses on classifying images of dogs, cats, and birds by implementing a baseline Convolutional Neural Network (CNN) and comparing its performance with advanced architectures, including VGG-16, ResNet variants, and EfficientNet (B4). This work is undertaken as part of a Kaggle competition[1]. The primary aim is to experiment, transfer learn, optimize and compare empirical findings and results for the stated problem using different data preparing approaches and model architectures.

The report is comprised of the following sections: Data and methodology, Results and discussion and Conclusion.

2 Data and Methodology

The data is comprised of train and test sets with 12,000 and 3,000 images, respectively; with all the images categorized under three labels: cat, dog and bird. The provided images are in 32x32 PNG format and are balanced across all three classes.

2.1 Data Loading, Augmentation and Splitting

This section will discuss the techniques used while data loading, approach, and results for the baseline.

The training images are resized to 32x32 pixels. They are further augmented by randomly rotating the images upto ± 15 degrees, randomly flipping horizontally, zooming in on the images randomly, applying random shear transformation of size 10 and distorting the perspective of the image. The brightness, contrast, saturation, and hue are also modified by ColorJitter using the values 0.2, 0.2, 0.2 and 0.1, respectively. And finally normalize the images using the precomputed mean and std dev of each channel. For the testing data, we limit the augmentation to simply resizing and then normalizing the images. After making stratified splits of train and validation, each of the datas of train, validation and test are loaded with baseline batchsize 32 while for pretrained models batchsize is 64. However it must be noted that for the pretrained models the images are resized as per expected specific input size of those models for e.g. 224 pixels for VGG, 380 pixels for EfficientNet, etc. Furthermore, all above augmentations are made on the training data; the validation and testing are just resized and normalized.

[1] Bilal Naseem. Dog vs Cat vs Bird. https://kaggle.com/competitions/dog-vs-cat-vs-bird, 2024. Kaggle.

2.2 Model Specification

2.2.1 Baseline CNN

The architecture used for CNN comprises of 4 convolutional blocks, each layer comprising of a conv layer with batch norm, ReLU, max pooling (2) and dropout (0.2). The number of filters for each convolutional layer are 32, 64, 128 and 256 in that order. The Fully Connected layers take in the fully connected input as 1024 and number of hidden layer neurons (just before the output layer) as 512. The FC layers again use the ReLU and dropout, while softmax is applied in the loss function.

For the training setup, we use the Cross Entropy Loss, Adam Optimizer with learning rate 0.001 and the scheduler as ReduceLROnPlateau with tuning set as patience of 2, factor of 0.5 and min lr as 0.00001. Other hyperparams such as L1 lambda were set as 0.0001, clip grad as 1 and epochs were set to 50.

However, the following combinations of configurations were explored:

- **Learning Rate**: (0.01, 0.001, 0.0001)
- **Minimum lr**: (0.0001, 0.00001)
- **Learning Rate Scheduler**: (ReduceLROnPlateau, CosineAnnealingLR)
- **L1 lambda**: (0.001, 0.0001, 0.00001)
- **Clip grad**: (0.5, 0.75, 0.9, 1.0, 2.0)
- **Epochs**: (20, 40, 50, 100)

For our comparative experiments, we employ VGG-16, ResNet and EfficientNet.

2.2.2 VGG-16

For VGG, our training data was augmented the same way with the difference of resizing to 224x224 pixels as per the pretrained model. The architecture was not changed except for modification to the classifier for which two approaches were explored.

In approach 1, VGG classifier is modified such that instead of mapping from 4096 units to 1000 units in final layer, another layer of 128 units is added instead of output of layer of 1000 units and finally an output of layer of 3 units is added at the end. So, $4096 > 1000$ to $4096 > 128 > 3$ where weights between 4096 to 128 to 3 units are trainable and rest are frozen.

For this approach, the following configurations were explored:

- **Learning Rate**: (0.001, 0.0001)
- **Minimum lr**: (0.0001, 0.00001)

We also chose the ReduceLROnPlateau as the Learning Rate Scheduler. To reduce redundancy, the results reported are of the best configuration which is lr=0.001 and minimum lr= 0.00001. The optimal hyperparameters from this approach and kept constant for the subsequent one

In approach 2 for the VGG classifier, instead of modifying the existing classifier new layers are added in the classifier and all of pretrained weights are kept frozen from the original VGG and only the newly added parameters are learned. So, $4096 > 1000$ to $4096 > 1000 > 128 > 3$ where weights between 1000 to 128 to 3 units are trainable and rest are frozen. This approach the same optimal hyperparameter configurations as that of Approach 1.

Since Approach 1 proved to be empirically better, we continued it for the remaining models.

2.2.3 ResNet

We further empirically test our Approach 1 used with VGG, now with ResNet. Considering dataset size and problem complexity following three ResNet Model Structures were explored:

- **ResNet-18**
- **ResNet-34**
- **ResNet-50**

We've not tested ResNet-101 and ResNet-152 considering small size of our training set. Plus, we can already see that VGG despite having higher number of parameters didn't necessarily offer exceptionally better performance. So the ideal approach for our dataset is to select a model which is deeper but still more efficient in terms of parameter usage. For ResNet-18, the following hyperparameter configuration was also explored:

- **Learning Rate**: (0.001, 0.0001)
- **Minimum lr**: (0.0001, 0.00001)

Again, only the selected configuration is shown to reduce redundancy. The best configuration of lr=0.001 and min lr= 0.00001 was kept constant for ResNet-34 and ResNet-50. For the final training the number of epochs are limited to 35 epochs as we're only transfer learning for the final layer and so the model should converge early. Also, the above VGG model revealed that 50 epochs are overkill as deeper models seem to converge in 20-35 epochs.

2.2.4 EfficientNet-B4

After ResNet, we could've tested DenseNet but we wanted to see if we could make a significant improvement over ResNet (accuracy $\geq 90\%$) so we skipped it and directly went to EfficientNet as it offers significantly better performance on ImageNet.

Also, we have only tested EfficientNet-B4. We have not tested smaller models keeping in view that maximum performance is the aim here and we have also not tested bigger models as we have already seen that increasing the parameters and using even more deeper network beyond a certain point doesn't work well with our dataset as we previously have opted for ResNet-34 over ResNet-50. EfficientNet-B4 is comparable in size to ResNet-34 in terms of number of parameters but more deeper so a better performance is expected!

This time we've experimented also with data preprocessing. We've tested if normalizing with mean and std calculated on ImageNet improves accuracy more than normalizing with mean and std calculated on training set. With our own training set being used for EfficientNet-B4, we apply the same data preprocessing and augmentation techniques; except we resize the images to 380x380 pixels. While the training after the preprocessing on the ImageNet dataset (where we use techniques like enlarging the shortest size to 400 and resizing the center to 380x380 pixels). It must be noted that for each pretrained model, the modified classifier layers used ReLU and dropout 20%.

2.3 Hyperparameter Optimization

The explored and optimal hyperparameters can be found under each model in the previous section. However, eventually the loss function, optimizer and LR Scheduler remained fixed throught the pretrained models which were CrossEntropyLoss, Adam and ReduceLROnPlateu. While L1 regularization and gradient clipping was only used in the baseline.

3 Results and Discussion

Figure 1: Baseline CNN results from the optimal hyperparameters configuration

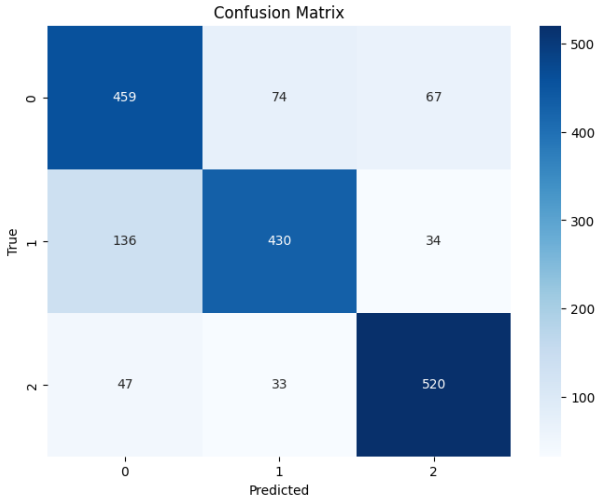

Figure 2: Baseline CNN, Confusion Matrix from Validation

For the baseline we report the results coming from the optimal hyperparams configuration on the validation set. The baseline results show the training loss to steadily decrease as epochs increase, while the validation loss and accuracy become almost constant after 30 epochs. The training accuracy follows an increasing trend overall reaching to 82%. However, the validation accuracy starts much higher than the training (almost 55% vs the training 45%) and improves much rapidly until stabilizing around 78% between 28 epochs till the end. As the training accuracy is greater than validation and the latter almost flattens after 30 epochs, we can safely imply that there is isn't overfitting in general. Furthermore, it should be noted that the following trend will prevail across baseline and all other models that the training loss will be higher than validation

loss with training accuracy being lower than its validation accuracy. This is because the training data was augmented while validation not.

As per the confusion matrix on the validation set in Figure 2 where 0, 1 and 2 are labelled as cat, dog, and bird respectively, we get the corresponding F1-scores as 0.74, 0.76 and 0.85. And the overall accuracy on validation as 78%.

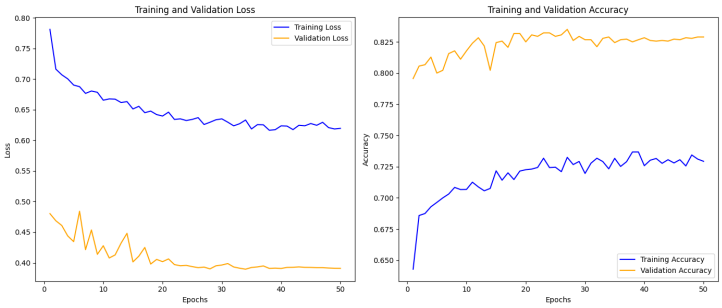

Figure 3: VGG-16 results from the optimal hyperparameters configuration

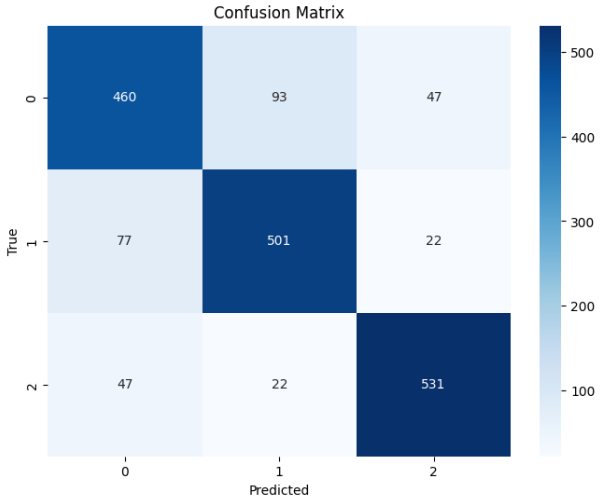

Figure 4: VGG-16, Confusion Matrix from Validation

The results for Approach 1 on the VGG show that the training loss decreases over the 10 epochs starting from 0.7817 to 0.6646. The validation loss behaves similarly starting at 0.6069 in Epoch 1 and decreases to 0.4267 in Epoch 10. Training accuracy also improves to 71% and the validation achieves even higher accuracy of 82.06%. The execution time for each epoch fluctuates between 104 and 105 seconds on average.

While the results of the second approach on the VGG show the training and validation losses to decrease overall, but with frequent fluctuations at different epochs. While the

training and validation accuracy increase from the epoch 1 to the last, again with slight and marginal fluctuations. The time also exceeds and takes about 106 seconds at a few epochs. After this empirical exploration, we finalize Approach 1 for VGG-16. Its confusion matrix from the validation set in Figure 4 reports an overall accuracy of 83% and F1-scores of 0.78, 0.82 and 0.89 for the cat, dog and bird classes, respectively.

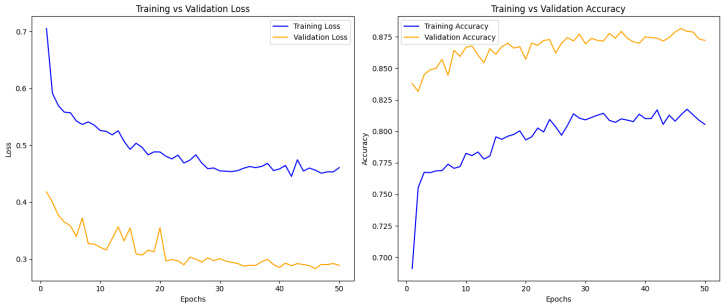

Figure 5: ResNet-34 results from the optimal hyperparameters configuration

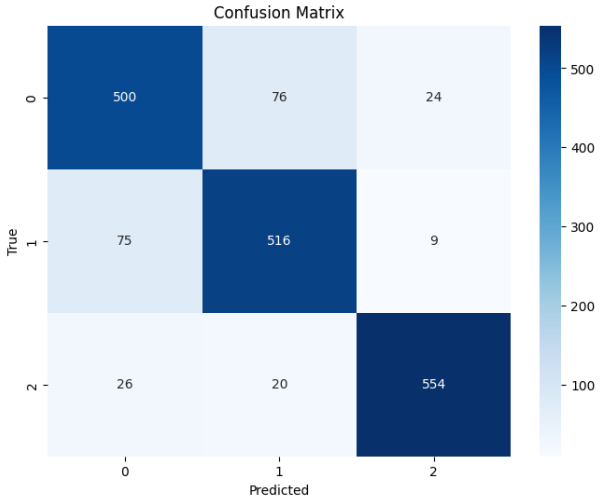

Figure 6: ResNet-34, Confusion Matrix from Validation

For ResNet-18, the training and validation accuracy at the end of 5 epochs are 75.77% and 82.44%, respectively and converging between 76-79 seconds. ResNet-34 performs slightly better as the training accuracy reaches just 76.83% and validation accuracy fluctuates but gains the highest accuracy of 84% (at Epoch 2) as compared to the other versions at similar epochs, taking about 80+ seconds. Furthermore, ResNet-50 takes 86+ seconds per epoch, and gives training and validation accuracies as 75.96% and 84.06%, respectively.

After this exploration, we find that ResNet-34 has overall lowest average validation losses. Those of ResNet-50 tend to diverge slightly faster in training and indicates it may be a higher risk of overfitting compared to ResNet-34. So by keeping in view the tradeoff, performance and execution time, we settle the final training with ResNet-34, as running a deeper model isn't yielding significantly better and optimal results. After running ResNet-34 for 35 epochs and then another 15 (50 epochs in total), we find that the validation accuracy plateaus to between 87 and 88%. The corresponding confusion matrix in Figure 6 reports the cat, dog and bird F1-scores to be 0.83, 0.85 and 0.93, respectively.

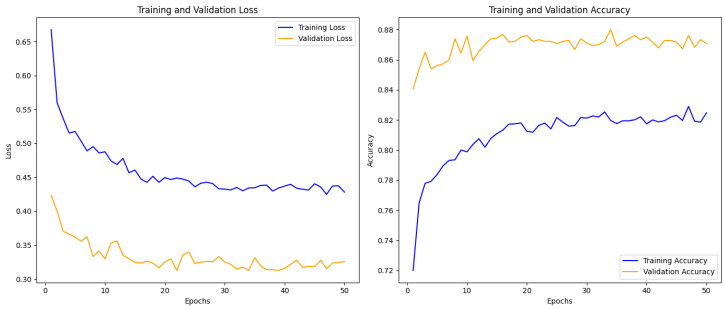

Figure 7: EfficientNet-B4 results from the optimal hyperparameters configuration

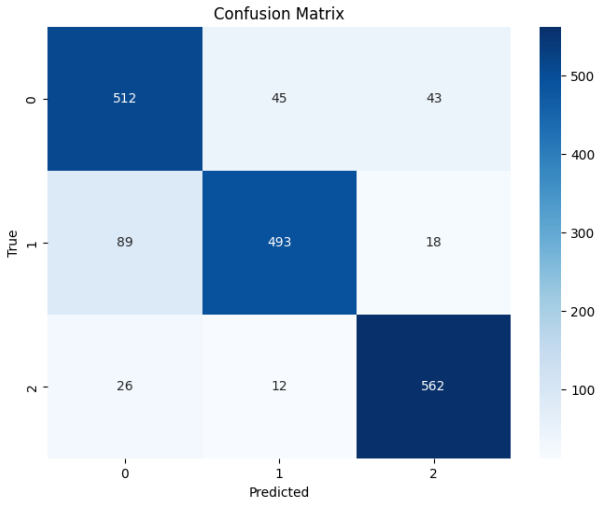

Figure 8: EfficientNet-B4, Confusion Matrix from Validation

For EfficientNet-B4, using our own training set preprocessing techniques and applying the same hyperparams as of ResNet final for 5 epochs, we see validation accuracy to reach 85.94% and the training and validation losses to rather stagnate.

The training after the preprocessing on the ImageNet dataset we find that our training-set normalization seems to result in slightly better validation accuracy. ImageNet normalization works reasonably well too, with a comparable performance in terms of training loss and accuracy. Both approaches are effective, but the training-set normalization might be better for our specific dataset, as it leads to slightly better and more consistent validation performance. The latter approach using ImageNet normalization gives a maximum validation accuracy of 84.56% but takes relatively more time for convergence than the former approach.

Initially we tried with lr=0.001, minimum lr=0.00001, and factor=0.5. While this configuration was good for the early stages of training but after 20 epochs model started to oscillate around 85% validation accuracy. So, we changed the minimum lr to 1e-6 and reduced factor to 0.3 to allow finer adjustments during late training with more gradual reductions in lr. The final approach of training with our training set normalization though fluctuates validation losses but yields a validation accuracy plateauing at around 87%. The following Figure 8 reports the F1-scores of the cat, dog and bird classes to be 0.83, 0.86 and 0.92, respectively.

If we compare the classification report to that of ResNet-34, there are minor differences in the regard that EfficientNet-B4 is more balanced w.r.t classes but overall, the performance is almost same as ResNet-34. This is contrary to our expectations as we expected the EfficientNet-B4 to have overall validation accuracy between 90% to 95%. A summary table of the each final pretrained model's classification report is as follows in Table 1. This result is again confirmed by identical performance of ResNet-34 and EfficientNet-B4 on the test set as found in Table 2.

Model	**Cat (F1-score)**	**Dog (F1-score)**	**Bird (F1-score)**	**Val Accuracy**
Baseline CNN	0.74	0.76	0.85	78%
VGG-16	0.78	0.82	0.89	83%
ResNet-34	0.83	0.85	0.93	87%
EfficientNet-B4	0.83	0.86	0.92	87%

Table 1: Comparison of F1-scores and validation accuracy across models

Model	**Test Accuracy**
Baseline CNN	77.60%
VGG-16	83.93%
ResBet-34	86.30%
EfficientNet-B4	86.30%

Table 2: Test Accuracy of Explored Models

4 Conclusion

While it seems possible to achieve 90% and beyond accuracy on test set with a bit more experimentation with models like ConvNeXt, RegNet, EfficientNetV2, BiT, ViTs, etc., the purpose for this study was to experiment and explore and optimize different models for our problem statement. Therefore, even though EfficientNet-B4 not going beyond the 86-87% accuracy is unusual, for the models used in this study the obtained results are as good as one can expect, and hence aim of the study is completed and we can iteratively continue applying more advance models to get higher and higher accuracy.