❖

ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING CHALLENGE 1

TEAM: LeCunTanh
Members:

- ❖ Fatih Temiz(10901682)
- ❖ Ecem İzde Kaya(10902759)
- ❖ Hessam Hashemizadeh(10900041)

IMAGE CLASSIFICATION CHALLENGE:

A) Data Preparation

For the given dataset which is [5200,96,96,3] sized, there are 3199 healthy and 2001 unhealthy images.

- ❖ Class weights: used for training for mitigating imbalanced dataset effects.Calculated according to frequency where most existing classes get less weight.[1]
  - ➢ Class 0 (Healthy): 0.80
  - ➢ Class 1 (Unhealthy): 1.31
- ❖ Outlier Analysis: When we manually inspected the data, we realized that there were two distinct outliers “Shrek" and "Trololo”.
  - ➢ 

  - ➢ The above images were filtered from data using their average_intensity(mean) where Shrek has 0.25(normalized) mean and Trololo 0.44(mean). There are 210 outliers in total filtered from the dataset.
- ❖ Resizing and Normalizing: Resizing (96,96,3) images to (224,224,3) and normalizing gives us higher accuracies with transfer learning models even though some of them have rescaling layers. We have also applied the same resizing to the "model.py" file to be coherent with the model during the submission.

B) Data Augmentation

Almost all data augmentation types tested below leveraged our model capabilities:

- ➔ RandomRotation
- ➔ RandomFlip
- ➔ RandomContrast
- ➔ RandomBrightness

However, RandomZoom and especially RandomTranslation does not provide improvement.

❖

Convolutional Neural Networks

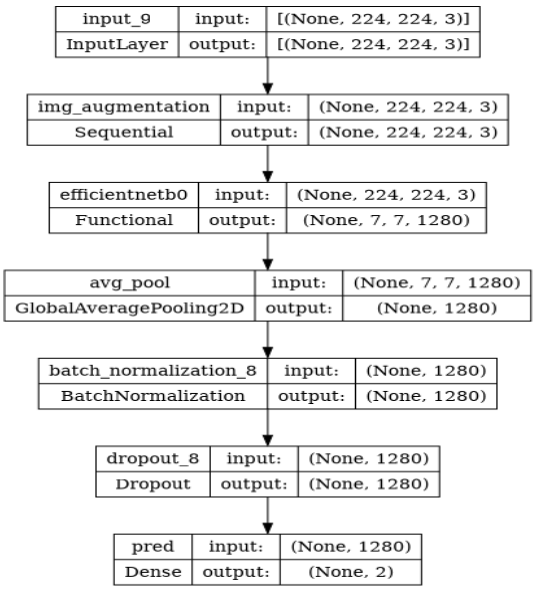
As a first step, we started a project using VGG16; we tried to develop our pipeline using VGG16. We trained VGG16 from scratch; we reached up to %87 accuracy on our test set with it. However, on CodaLab we reached at most %70 using this model. From this point, we decided to use transfer learning. We worked with MobileNetV2 & EfficientNetB0 and kept our detailed work in EfficientNetB0 later.

VGG16: At this point; we were using one neuron at the output with sigmoid function; we realized that it was not sufficient to predict the test set since we obtained %68(max). Here we also applied k-fold cross-validation to see more robust results in terms of accuracy [4]

Transfer Learning:

There are quite well-known models that were previously trained on the ImageNet dataset. It is proved that these models can be used to extract features with pre-trained weights. Among various models, we implemented MobileNetV2 and EfficientNetB0[2,6]. Later, we observed that EfficientNetB0 shows at least %3 higher accuracy on all hyperparameter combinations. From this point, EfficientNetB0 is used as the base model.

- **Optimizers:**
  - **SGD:** We have observed that it has very slow convergence and it is prone to overfit more than Adam.
  - **RMSProp:** RMSProp is not recommended with EfficientNetB0[1]
  - **Adam**: We have applied Adam in the feature extraction phase with a high learning rate when all EfficientNetB0 layers were frozen.
  - **AdamW**: We have applied AdamW in the fine-tuning phase since we unfreeze the top 50 layers with it; it has the resiliency to overfit since it uses regularization.[2]
- Using 2 neurons at output with one-hot encoding, we obtained better results; our CNN architecture is as follows.



❖

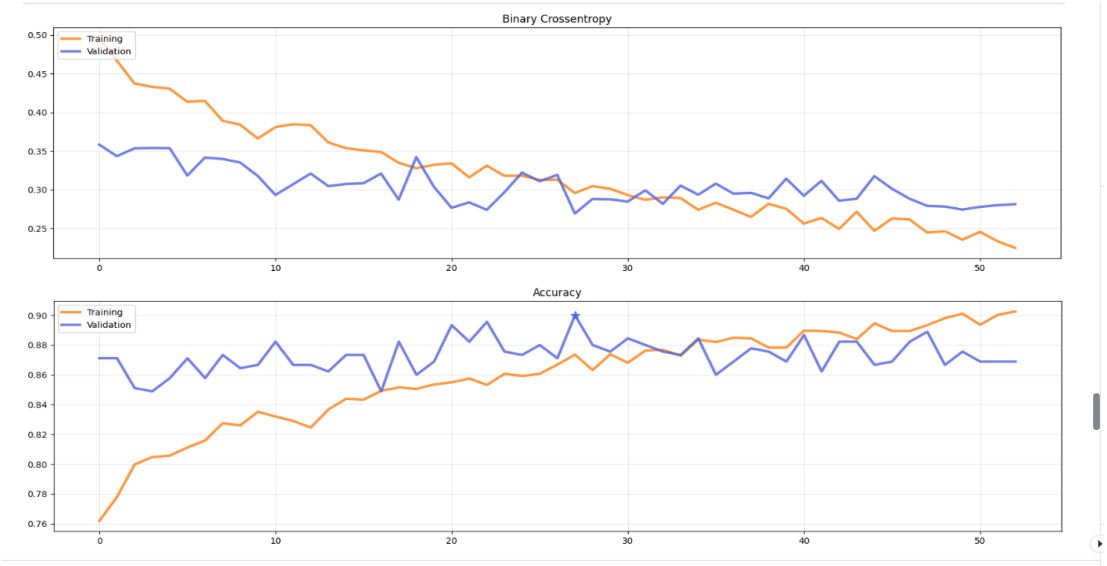
Fine Tuning

After Transfer Learning with EfficientNetB0, we have obtained almost %85 test set accuracy. However, we first unfreeze the top 20 layers as suggested[1]. In Phase 1 of the competition, we obtained %87 accuracy with this model, however, we obtained %76.3 in Phase 2. Unfreezing all layers results in overfitting of the model on training data. We obtained %93,75 accuracy on our test set by unfreezing all layers however we obtained %76.3 again on the second phase of codalab again. Finally, we obtained %79.3(our best accuracy in the second phase) using the top 50 layers unfrozen. BatchNormalization Layers are kept frozen in the fine-tuning scheme. Otherwise, the updates applied to the non-trainable weights will suddenly destroy what the model has learned.[1,3]

Hyperparameters

Our best model(%87 accuracy in phase 1(just #of Unfrozen layer was 20 differently), %79.3 accuracy in phase 2 has the following hyperparameters and Cross-entropy and Accuracy plot.

Batch Size	32
Epochs_TransferLearning	50
Epochs_FineTuning	50
Optimizer_TransferLearning	Adam(lr=1e-3)
Optimizer_FineTuning	AdamW(lr=1e-4,wd=5e-4)
Callbacks	EarlyStop(patience=20)
# of Unfrozen Layer	50



Conclusion: We have conducted several implementations, using relatively small epochs(40-50) gave always better results in terms of unseen test accuracy. As a further step, larger models can be implemented such as ConvNextLarge[7]. Transfer Learning, Image Augmentation, Outlier Analysis, and Class Weights were leveraging concepts in our study.

❖

References

[1]:https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/
[2]:https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/
[3]:https://towardsdatascience.com/why-adamw-matters-736223f31b5d
[4]:https://www.kaggle.com/code/brianstumph846/efficientnet-b4-with-frozen-batchnorm2d-btll
[5]:https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide
[6]:https://blog.roboflow.com/how-to-train-mobilenetv2-on-a-custom-dataset/
[7]:https://medium.com/augmented-startups/convnext-the-return-of-convolution-networks-e70cbe8dabcc

Contributions:

All group worked together, and each member has equal contribution on project.