MultiNet: Binary Image Classifier

Taiel Maximiliano Pose Marino, Khayotbek Kamchiev
2045940, 1987223

December 7, 2024

1 Introduction

In the realm of machine learning, particularly in image classification tasks like the one we have now of classifying if an image contains fire or not, the balance between model complexity and performance is crucial and must be considered depending not only on the task at hand and the hardware constraints but also on the final objective. For us, the objective isn't to obtain the most balanced model nor the simplest one but to attain the highest accuracy we can with the resources we have available. For a task as simple as the one we have the difference between a CNN and a ResNet model is no bigger than a few percentage points, it is because of this that we explore the implementation of an ensemble model which we call MultiNet. The before-mentioned model leverages three ResNet architectures to improve binary classification accuracy without significantly increasing model size, since that would prevent us from training the models on consumer accessible hardware.

2 Methodology

2.1 Individual ResNet Models

We initiated our approach by training three ResNet models independently:

- **ResNet50**: Trained locally due to its smaller size and computational requirements.
- **ResNet152**: Trained on Kaggle's cloud infrastructure.
- **ResNet304**: A custom model where we doubled the layers of ResNet152, also trained on Kaggle.

2.2 Ensemble Model

After training the resnet models, their outputs were fed into a Convolutional Neural Network (CNN) with four fully connected layers, followed by a sigmoid

function to output a probability between 0 and 1, indicating the certainty of classification. Since all the models didn't fit on a single gpu we took advantage of the dual t4s offered by kaggle to train this final classifier.

2.3 CNN Model

A secondary CNN model was also implemented to help the MultiNet model reach higher accuracy. This CNN model contains three convolutional layers, each followed by Relu activation and pooling, to extract features from input images. These features are then flattened and processed through six fully connected layers, with dropout applied after each layer to prevent over-fitting.

2.4 Final Decision Making

The original model plateaued at a bit under 99% so in order to improve the accuracy of MultiNet we decided to train the secondary CNN to help it assign a class to the images it was unsure of. The threshold for uncertainty was set at 15%. If it wasn't reached the output of the MultiNet model was compared with the prediction of the CNN model, with a decision mechanism in place to choose based on the confidence levels of both models.

3 Results

- **Validation Accuracy**: Post-ensemble, our model achieved around 98.5% accuracy.
- **Combined Model Accuracy**: With the integration of secondary CNN, preliminary results suggest that accuracy could exceed 99%.

The following figures illustrate the training and validation metrics of our models:

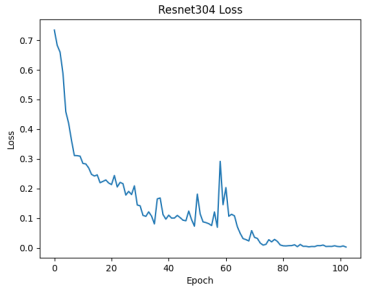


Figure 1: ResNet304 Loss

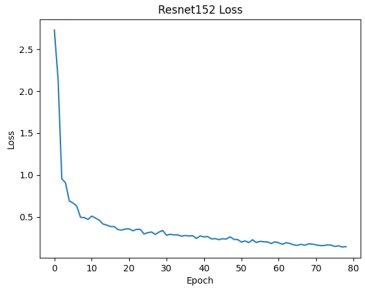
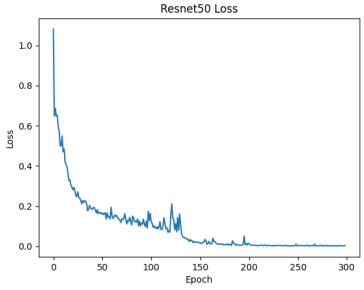
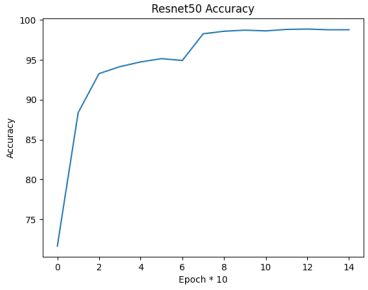


Figure 2: ResNet152 Loss

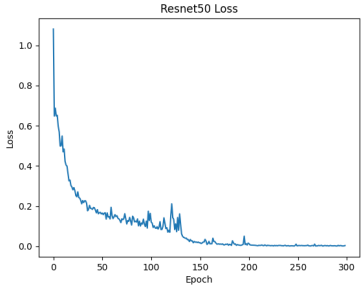


(a) ResNet50 Loss

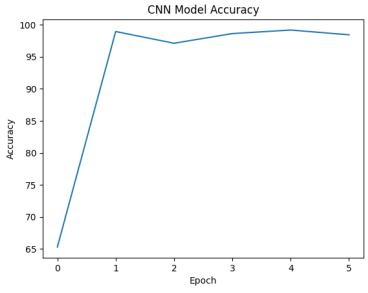


(b) ResNet50 Accuracy

Figure 3: ResNet50 Metrics



(a) CNN Loss



(b) CNN Accuracy

Figure 4: CNN Metrics

4 Analysis

The plot of the loss of all models indicate a decreasing trend as epochs increase, with some fluctuations in resnet304, resnet50, and in the CNN. The accuracy of all models quickly plateaus near 95%, showing that if a few percentage point difference doesn't cause many issues similar effectiveness and stability in classification can be reached with much smaller models. For example, the CNN reached 95% accuracy on the validation set in only 25 minutes compared to the hours spent in training all the individual models in MultiNet plus the couple of hours later needed to train the decision making CNN.

5 Conclusion

The MultiNet approach, combining multiple ResNet architectures and a decision-making CNN, demonstrates potential for high accuracy in binary image classification tasks. Further optimization, particularly in model integration and decision thresholds, could yield even higher performance metrics, it's also worth mentioning that if the complete automatization of the classification isn't necessary it's possible to manually analyze the results for which the model is less confident (i.e. confidence less than .02) which would further increase the accuracy of the predictions.

6 Architectures

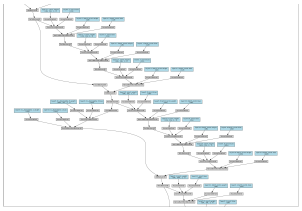


Figure 5: Partial Resnet304 Architecture

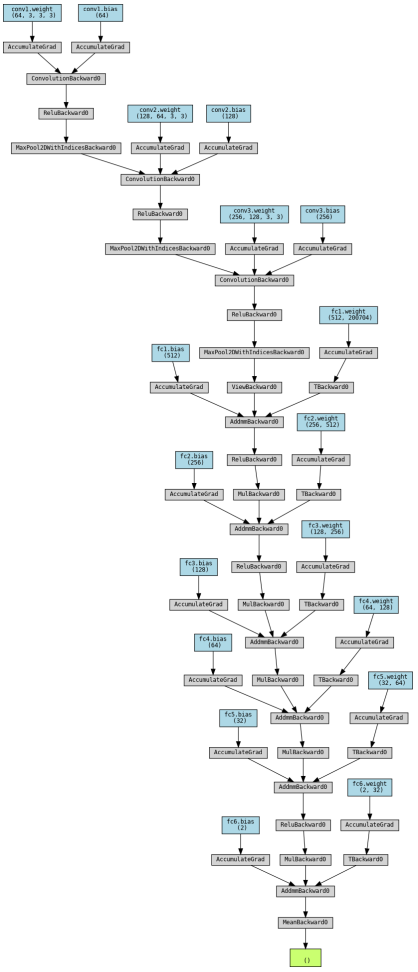


Figure 6: CNN Architecture

7 References

- **Github Repository**: https://github.com/poset26/DL_Challenge_BinaryClassifier
- **ResNEt**: https://github.com/JayPatwardhan/ResNet-PyTorch/tree/master
- **Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition**: https://arxiv.org/pdf/1512.03385.pdf