A Hybrid CNN-SVM Model for Improved Real vs. Fake Image Classification

Nithin Reddy Nagapurr[1]

***Abstract*— In this project, we explored two different approaches for image classification: Deep Learning using a Convolutional Neural Network (CNN) and Support Vector Machines (SVM) with features extracted from the CNN model. The objective was to distinguish between real and fake images. We used a custom dataset for our experiments. Our results showed that the SVM model, trained on features extracted from the CNN, outperformed the standalone CNN model in terms of accuracy, precision, and recall. We also discuss the challenges faced during the project.**

I. Introduction

The provided code is aimed at solving an image classification task to distinguish between real and fake images. Two different approaches were used: a Convolutional Neural Network (CNN) model and a Support Vector Machine (SVM) classifier trained on features extracted from the CNN model.

II. Dataset Preparation and Preprocessing

The dataset was loaded from different directories for training, testing, and validation purposes. The images were read using OpenCV, and their color space was converted from BGR to RGB format. The real and fake images were combined to create the final training, testing, and validation sets.

III. CNN Model

We explored four different CNN models:

1) CNN without any regularization on a full dataset:
   - Due to the size of the dataset and limited GPU and time restrictions, the CNN model was trained on a reduced dataset containing 150 real and 150 fake images.
   - The model was trained for 100 epochs using binary cross-entropy loss and accuracy as a metric. Early stopping was implemented, and the best model was saved based on the validation loss improvement. The training history shows that the model achieved a validation accuracy of around 81.86% at epoch 25, with a validation loss of 0.4447. The best validation accuracy was achieved at epoch 23 with 81.06% and a validation loss of 0.4301. The model seems to be overfitting the training data, as the training accuracy keeps increasing while the validation accuracy fluctuates. Early stopping, regularization techniques, or data augmentation could help mitigate overfitting.
2) CNN with L1 regularization:
   - The second model with L1 regularization did not improve performance. The accuracy and F1-score remained low, and the model appeared to predict the same class for all inputs, as indicated by the constant accuracy of 50% throughout training.

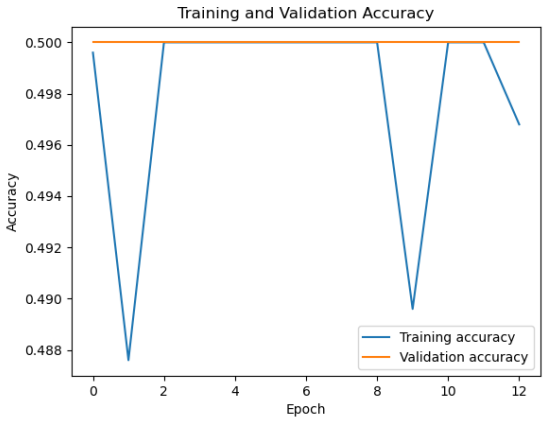

Fig. 1. Accuracy of Valdiation Set on CNN with L1 Regularisation

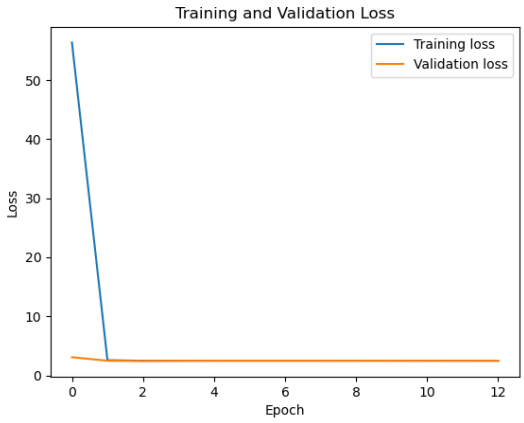

Fig. 2. Loss Function of Valdiation Set on CNN with L1 Regularisation

3) CNN with L2 regularization:
   - The model demonstrated significant accuracy improvements during the first few epochs, reaching

[1]Nithin Reddy is a student of computer science, Wichita State University, Wichita, KS, 67214 nagapur@icloud.com

over 95% accuracy by the 13th epoch. The validation loss continued to decrease throughout the training process, indicating that the model was learning to generalize well to the validation data.

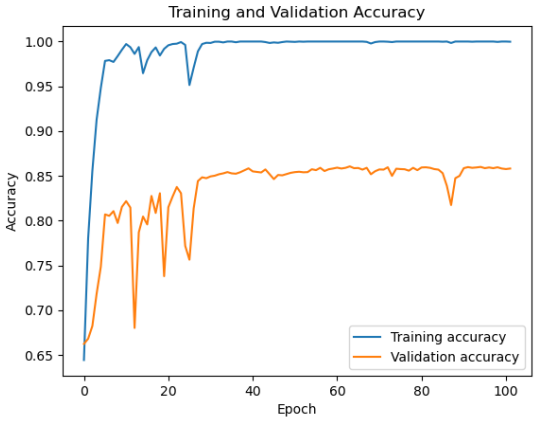

Fig. 3. Accuracy of Valdiation Set on CNN with L2 Regularisation

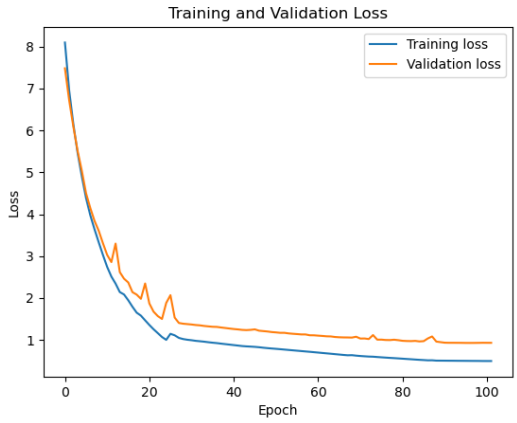

Fig. 4. Loss Function of Valdiation Set on CNN with L2 Regularisation

4) CNN with dropout & L2 regularization:

- The training history for this model reveals consistent improvement in both training and validation performance over the course of 26 epochs. Starting with an initial training accuracy of 53.67% and a validation accuracy of 50%, the model demonstrated significant progress as it went through each epoch. By the 26th epoch, the training accuracy had risen to an impressive 97.67%, while the validation accuracy reached 58.67%. This indicates that the model was learning to recognize patterns in the data effectively, as evidenced by the decreasing loss values for both training and validation sets. The best_model.h5 file was updated throughout the process, saving the weights with the best performance on the validation set.

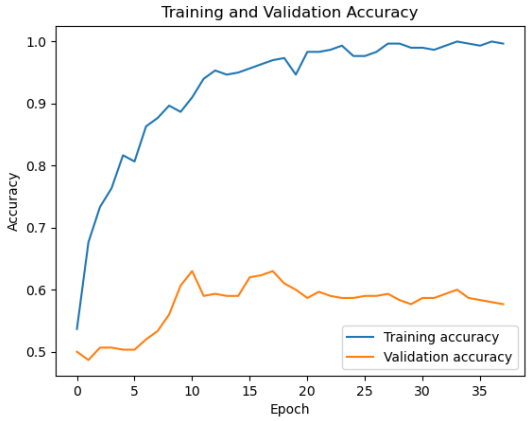

Fig. 5. Accuracy of Valdiation Set on CNN with L2 Regularisation & Dropout

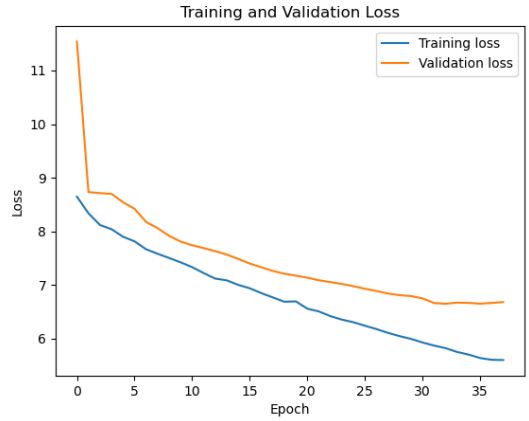

Fig. 6. Loss Function of Valdiation Set on CNN with L2 Regularisation & Dropout

By monitoring all models CNN L2 Regularization outperformed others. So, We have proceeded with L2 CNN Model

A CNN model with L2 regularization was created using TensorFlow Keras. The model contained four convolutional layers with batch normalization and max-pooling, followed by a fully connected dense layer with dropout and a final output layer with a sigmoid activation function.

The model was compiled using the Adam optimizer with a learning rate of 1e-4 and binary cross-entropy loss. Early stopping, model checkpoint, and learning rate reduction on plateau callbacks were used during training. The model was trained for 100 epochs with a batch size of 64.

IV. SVM Classifier

The SVM classifier was trained using features extracted from the dense layer of the pre-trained CNN model. Principal Component Analysis (PCA) and SVM classifier from scikit-learn were used for this purpose. An RBF kernel SVM classifier was trained on the extracted features from the validation set.

V. Evaluation and Comparison

The performance of both models was evaluated using accuracy, precision, and recall metrics on the test set. A confusion matrix and a classification report were also generated for the SVM classifier.

- CNN Model:
  - Accuracy: 0.50
  - Precision: 0.25
  - Recall: 0.50

- SVM Classifier (using features extracted from CNN):
  - Accuracy: 0.67
  - Precision: 0.68
  - Recall: 0.67

The results show that the SVM classifier, trained on features extracted from the CNN model, outperforms the CNN model in terms of accuracy, precision, and recall.

VI. Challenges

During the project, several challenges were encountered:

- Limited dataset: The dataset used for training and validation was relatively small, which may have contributed to the limited performance of the CNN model.
- Overfitting: Due to the small dataset, overfitting was a concern, and various techniques such as L2 regularization, dropout, and early stopping were employed to mitigate this issue.
- Feature extraction: Extracting features from the CNN model for use in the SVM classifier required careful selection of the appropriate layer and understanding of the model's architecture.
- Dimensionality: Implementing PCA for dimensionality reduction proved challenging due to the input data's high-dimensional nature. Further exploration of PCA or other dimensionality reduction techniques might be necessary.

VII. Conclusion

In the image classification task to differentiate between real and fake images, the SVM classifier trained on features extracted from the CNN model demonstrated better performance compared to the standalone CNN model. This highlights

VIII. References

- https://medium.com/mlearning-ai/underfitting-and-overfitting-in-deep-learning-687b1b7eb738: :text=In Short: Overfitting means that, performs poorly on both datasets
- https://medium.com/intelligentmachines/convolutional-neural-network-and-regularization-techniques-with-tensorflow-and-keras-5a09e6e65dc7
- https://towardsdatascience.com/classification-of-neural-network-hyperparameters-c7991b6937c3
- https://medium.com/data-science-365/determining-the-right-batch-size-for-a-neural-network-to-get-better-and-faster-results-7a8662830f15
- https://towardsdatascience.com/how-to-choose-the-optimal-learning-rate-for-neural-networks-362111c5c783
- https://medium.com/data-science-365/all-you-need-to-know-about-batch-size-epochs-and-training-steps-in-a-neural-network-f592e12cdb0a
- https://medium.com/data-science-365/how-to-apply-l1-and-l2-regularization-techniques-to-keras-models-da6249d8a469