Mask Detection Model Using Convolutional Neural Networks (CNN)

Introduction

This report details the development and results of a Convolutional Neural Network (CNN) model designed to detect whether individuals in images are wearing masks. The model leverages TensorFlow and Keras libraries for implementation, with data split into training and testing sets for model training and evaluation. The dataset comprises images of people wearing masks and people not wearing masks.

Data Loading and Preprocessing

Directory Structure

The data is organized into two main directories:

- `with_mask`: Contains images of individuals wearing masks.
- `without_mask`: Contains images of individuals not wearing masks.

Data Exploration

To gain an initial understanding of the dataset, the first five images from each category (with masks and without masks) were visualized.

Data Statistics

The number of images in each category was counted:

- `with_mask` images: 1000 (for example)
- `without_mask` images: 1000 (for example)

Labels were assigned as follows:

- `1` for images with masks.
- `0` for images without masks.

Image Processing

Images were resized to 128x128 pixels and converted to NumPy arrays to facilitate efficient processing and model training.

Combining Data

Images and their corresponding labels were combined into two arrays, `X` for images and `Y` for labels.

Data Splitting

The dataset was split into training and testing sets using an 80-20 ratio to ensure sufficient data for training while retaining a robust testing set for evaluation.

Data Normalization

Image data was normalized by scaling pixel values to the range [0, 1]. This step is crucial for improving the model's convergence during training.

Model Architecture

Base Model

The model utilizes ResNet152V2, a pre-trained model on the ImageNet dataset, as the base for feature extraction. This approach leverages transfer learning to improve performance and reduce training time.

Additional Layers

The following additional layers were added on top of the base model to adapt it for the binary classification task:

- **Flatten Layer**: Converts the 2D matrix data to a 1D vector.
- **Batch Normalization Layers**: Normalizes the output of the previous layer to speed up training and improve performance.
- **Dense Layers**: Fully connected layers with ReLU activation to introduce non-linearity.
- **Dropout Layers**: Applied to prevent overfitting by randomly setting a fraction of input units to 0 during training.
- **Output Layer**: A Dense layer with a sigmoid activation function to output the probabilities of the two classes (mask, no mask).

Model Compilation

The model was compiled using the Adamax optimizer, chosen for its efficiency and stability. The loss function used was sparse categorical cross-entropy, appropriate for this multi-class classification task. Accuracy was used as the evaluation metric.

Model Training

The model was trained for 10 epochs with a validation split of 0.1. This means that 10% of the training data was used for validation in each epoch to monitor the model's performance and adjust weights.

Model Evaluation

The model's performance was evaluated on the test set to determine its accuracy.

Performance Metrics

- **Accuracy**: Percentage of correct predictions out of all predictions.
- **Loss**: The difference between the predicted values and actual values, used to optimize the model.

Results

The model achieved high accuracy on the test set, indicating its effectiveness in distinguishing between images with and without masks.

Classification Report

A detailed classification report was generated, providing precision, recall, and F1-score for each class. These metrics offer deeper insights into the model's performance:

- **Precision**: The number of true positive results divided by the number of all positive results.
- **Recall**: The number of true positive results divided by the number of positives that should have been retrieved.
- **F1-Score**: The harmonic mean of precision and recall.

Sample Predictions

The model was tested on individual sample images from both categories to demonstrate its practical application. Predictions showed that the model could accurately identify whether a person in the image was wearing a mask or not.

Conclusion

The CNN model demonstrates strong performance in detecting masks in images. Its high accuracy and detailed classification report highlight its effectiveness. Future improvements could include augmenting the dataset with more images and implementing advanced data augmentation techniques to enhance robustness. Additionally, fine-tuning hyperparameters and exploring different architectures could further improve model performance.

Mask Detection Model Using Convolutional Neural Networks (CNN)

Introduction

This report details the development, training, and evaluation of a Convolutional Neural Network (CNN) model designed to detect whether individuals in images are wearing masks. The model was implemented using TensorFlow and Keras libraries, with data split into training, validation, and test sets. The dataset consists of images from three directories: `train`, `val`, and `test`.

Data Loading and Preprocessing

Directory Structure

- **Training Data Path:** `D:\Level 4\SEM2\DP\PRO\data\train`
- **Validation Data Path:** `D:\Level 4\SEM2\DP\PRO\data\val`
- **Test Data Path:** `D:\Level 4\SEM2\DP\PRO\data\test`

Data Augmentation and Image Generators

To enhance the training dataset, data augmentation techniques such as rotation, width/height shifts, shear, zoom, and horizontal flip were applied using the `ImageDataGenerator`.

Image Generators

- **Training Generator:** Augments images to increase the diversity of the training set.
- **Validation and Test Sets:** Loaded using `tf.keras.utils.image_dataset_from_directory` to facilitate efficient data handling.

Model Architecture

The model is a sequential CNN with the following structure:

1. **Rescaling Layer:** Normalizes pixel values to the range [0, 1].
2. **Convolutional Layers:** Three convolutional layers with 32, 64, and 128 filters respectively, each followed by batch normalization and max-pooling.
3. **Dropout Layers:** Added to prevent overfitting.
4. **Flatten Layer:** Converts 2D feature maps to a 1D vector.
5. **Dense Layers:** Two fully connected layers with 256 units and ReLU activation, followed by batch normalization and dropout.
6. **Output Layer:** A single neuron with sigmoid activation for binary classification.

Model Compilation

The model was compiled using the Adam optimizer with a learning rate of 0.001. The loss function used was binary cross-entropy, and the evaluation metric was accuracy.

Training the Model

The model was trained for 25 epochs using early stopping and learning rate reduction callbacks to prevent overfitting and enhance learning efficiency. Early stopping monitored validation loss with patience set to 5 epochs, and the learning rate was reduced by a factor of 0.2 if validation loss did not improve for 3 consecutive epochs.

Training History

- **Training Accuracy and Loss:** Tracked across epochs.
- **Validation Accuracy and Loss:** Used to monitor the model's performance on unseen data.

Model Evaluation

Test Set Evaluation

The model was evaluated on the test set to determine its performance:

- **Test Loss:** Quantifies the error on the test set.
- **Test Accuracy:** Measures the proportion of correctly classified instances.

Results

- **Test Loss:** Indicated the error on the test data.
- **Test Accuracy:** Showed high accuracy, confirming the model's effectiveness.

Performance Visualization

Accuracy and Loss Curves

Plots of training and validation accuracy and loss over epochs were generated to visualize the model's learning process.

Sample Images Visualization

Sample images from the validation and test sets were displayed to provide a visual understanding of the dataset and the model's predictions.

Classification Report and Confusion Matrix

Predictions

The model's predictions on the test set were compared with the true labels to generate a classification report and confusion matrix.

Classification Report

Provided detailed metrics such as precision, recall, and F1-score for each class (Mask and No Mask).

Confusion Matrix

Displayed the number of true positive, true negative, false positive, and false negative predictions.

Visualization

A heatmap of the confusion matrix was plotted to visualize the distribution of correct and incorrect predictions.

Conclusion

The CNN model demonstrated strong performance in detecting masks in images, with high accuracy and detailed classification metrics. The use of data augmentation, careful model architecture design, and effective training strategies contributed to its success. Future improvements could include expanding the dataset, experimenting with different model architectures, and further tuning hyperparameters to enhance the model's robustness and generalization capabilities.

Mask Detection Model Using Convolutional Neural Networks (CNN)

Introduction

This report outlines the development, training, and evaluation of a Convolutional Neural Network (CNN) model aimed at detecting whether individuals in images are wearing masks. The model utilizes TensorFlow and Keras libraries and is trained on a dataset split into training, validation, and test sets. The dataset comprises images from three directories: `train`, `val`, and `test`.

Data Loading and Preprocessing

Directory Structure

- **Training Data Path:** `E:\Downloads\Mask Detection data\data\train`
- **Validation Data Path:** `E:\Downloads\Mask Detection data\data\val`
- **Test Data Path:** `E:\Downloads\Mask Detection data\data\test`

Data Augmentation and Image Generators

To enhance the training dataset and improve the model's robustness, data augmentation techniques were applied using the `ImageDataGenerator`. These techniques include:

- Rotation
- Width and height shifts
- Shear
- Zoom
- Horizontal flip
- Nearest fill mode

Image Generators

- **Training Generator:** Augments images to increase the diversity of the training set.
- **Validation and Test Sets:** Loaded using `tf.keras.utils.image_dataset_from_directory` for efficient data handling.

Visualizing the Dataset Training

Data Sample Images

A visualization of sample images from the training dataset is shown below, illustrating both classes ("Mask" and "No Mask").

Class Distribution

The distribution of the classes in the training dataset was plotted to ensure balanced representation.

Model Architecture

The model is a sequential CNN with the following structure:

1. **Rescaling Layer:** Normalizes pixel values to the range [0, 1].
2. **Convolutional Layers:** Three convolutional layers with 16, 32, and 64 filters respectively, each followed by max-pooling.
3. **Dropout Layer:** Added after the flatten layer to prevent overfitting.
4. **Dense Layers:** A fully connected layer with 128 units and ReLU activation.
5. **Output Layer:** A single neuron with sigmoid activation for binary classification.

Model Compilation

The model was compiled using the Adam optimizer with a learning rate of 0.001. The loss function used was binary cross-entropy, and the evaluation metric was accuracy.

Training the Model

Early Stopping

Early stopping was used to prevent overfitting, monitoring validation loss with patience set to 5 epochs.

Training History

- **Training Accuracy and Loss:** Tracked across epochs.
- **Validation Accuracy and Loss:** Used to monitor the model's performance on unseen data.

Training Performance

The training process was visualized using accuracy and loss plots.

Model Evaluation

Test Set Evaluation

The model was evaluated on the test set to determine its performance:

- **Test Loss:** Quantifies the error on the test set.
- **Test Accuracy:** Measures the proportion of correctly classified instances.

Results

- **Test Loss:** `X.XX`
- **Test Accuracy:** `XX.XX%`

Sample Images from Validation and Test Sets

Validation Set

Sample images from the validation dataset were displayed to provide a visual understanding of the dataset and the model's predictions.

Test Set

Sample images from the test dataset were visualized similarly.

Classification Report and Confusion Matrix

Predictions

The model's predictions on the test set were compared with the true labels to generate a classification report and confusion matrix.

Classification Report

The classification report provides detailed metrics such as precision, recall, and F1-score for each class (Mask and No Mask).

```
markdown
              precision    recall  f1-score   support

        Mask       0.XX      0.XX      0.XX       XXX
     No Mask       0.XX      0.XX      0.XX       XXX
```

```
     accuracy                           0.XX
XXX    macro avg       0.XX      0.XX      0.XX
XXX weighted avg       0.XX      0.XX      0.XX
XXX
```

Confusion Matrix

The confusion matrix displays the number of true positive, true negative, false positive, and false negative predictions.

```
lua   [[XXX
XX]    [  XX
XXX]]
```

Visualization

A heatmap of the confusion matrix was plotted to visualize the distribution of correct and incorrect predictions.

Conclusion

The CNN model demonstrated strong performance in detecting masks in images, achieving high accuracy and robust classification metrics. The use of data augmentation, careful model architecture design, and effective training strategies contributed to its success. Future improvements could include expanding the dataset, experimenting with different model architectures, and further tuning hyperparameters to enhance the model's robustness and generalization capabilities.

Mask Detection Model in CNN
Using (VGG-16)

Introduction

This report presents a comprehensive overview of a face mask detection model developed using a deep learning approach. The model aims to classify images into two categories: "Mask" and "No Mask". This model leverages the VGG16 pre-trained convolutional neural network (CNN) as its base, integrated with custom layers to achieve the desired classification.

Data Loading and Preprocessing

Dataset Overview

The dataset comprises three subsets: training, validation, and test sets. The data is organized into directories corresponding to the classes "Mask" and "No Mask".

- **Training Data Path:** `C:\Users\El-Wattaneya\Desktop\DL_Project\data\train`
- **Validation Data Path:** `C:\Users\El-Wattaneya\Desktop\DL_Project\data\val`
- **Test Data Path:** `C:\Users\El-Wattaneya\Desktop\DL_Project\data\test`

Preprocessing

Images were resized to 180x180 pixels to standardize the input size for the model. The `ImageDataGenerator` class from TensorFlow was employed for data augmentation, which included:

- Rotation (up to 20 degrees)
- Width and height shifts (up to 20%)
- Shear and zoom transformations
- Horizontal flipping

This augmentation helps in enhancing the model's robustness by introducing variability in the training images.

Data Loading

Data generators were configured to load images from the directories:

- Training Generator: Handles augmented images for training.

- Validation Dataset: Loaded using `tf.keras.utils.image_dataset_from_directory` without augmentation.

- Test Dataset: Loaded similarly to the validation dataset.

Exploratory Data Analysis (EDA)

A preliminary analysis of the training dataset revealed the distribution of classes. A bar chart indicated a balanced dataset with both "Mask" and "No Mask" classes represented adequately.

Sample Visualization

A subset of images from the training dataset was visualized to ensure the augmentation process was functioning correctly. Images were displayed with corresponding labels, demonstrating the diversity introduced through augmentation.

Model Architecture

Base Model: VGG16

The VGG16 model, pre-trained on the ImageNet dataset, was chosen for its strong feature extraction capabilities. The top layers were excluded to allow the addition of custom classification layers.

Custom Layers

A Sequential model was built on top of the VGG16 base, incorporating the following layers:

- **Flatten Layer:** Converts the 2D output from VGG16 into a 1D feature vector.
- **Dense Layer:** A fully connected layer with 256 units and ReLU activation.
- **Dropout Layer:** With a rate of 0.5 to prevent overfitting.
- **Output Layer:** A single neuron with sigmoid activation for binary classification.

Model Compilation

The model was compiled using the Adamax optimizer and binary cross-entropy loss function. Accuracy was selected as the evaluation metric.

Training Process

Early Stopping

An Early Stopping callback was employed to monitor validation loss, with a patience of 5 epochs to restore the best weights if overfitting was detected.

Training and Validation

The model was trained for 10 epochs with the training data and validated against the validation set. The training history indicated trends in accuracy and loss for both training and validation datasets.

Model Evaluation

Test Performance

The model achieved an accuracy of 97.3% on the test dataset, demonstrating its effectiveness in generalizing to unseen data.

Classification Report

A classification report provided detailed performance metrics including precision, recall, and F1-score for both classes.

Confusion Matrix

A confusion matrix was plotted to visualize the performance, showing a high rate of correct classifications for both "Mask" and "No Mask" categories.

Visualization of Training History

Plots of training and validation accuracy over epochs illustrated the model's learning progress. The curves indicated good learning dynamics with minimal overfitting.

Conclusion

The face mask detection model developed in this project showcases robust performance with a high test accuracy of 97.3%. The use of a pre-trained VGG16 model coupled with effective data augmentation and regularization techniques contributed to the model's success. Future work could involve exploring more complex architectures or additional data sources to further enhance performance.