Machine Learning Class Project

Pikachu Image Classification using Machine Learning

Zhengliang Jiang
May 22, 2024

Agenda

1) Project Overview
2) Data Preprocessing & EDA
3) Model Architecture
4) Model Evaluation
5) Improve the Model
6) Conclusion

Project Overview

- ➢ **Project Background:** The aim of this project is to leverage machine learning to classify images of Pikachu. Pikachu is a well-known character from the Pokémon franchise, characterized by its yellow color and unique shape.

- ➢ **Scope:** The project involves building a model to distinguish between images of Pikachu and other objects. The dataset used contains labeled images categorized into 'Pikachu' and 'Not Pikachu’.

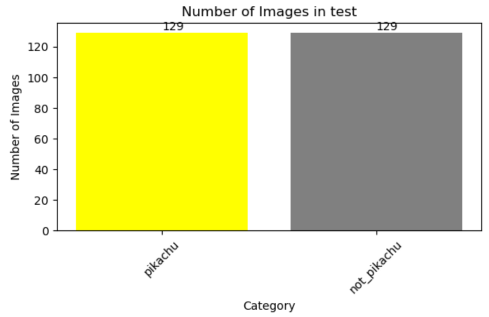
Project Overview

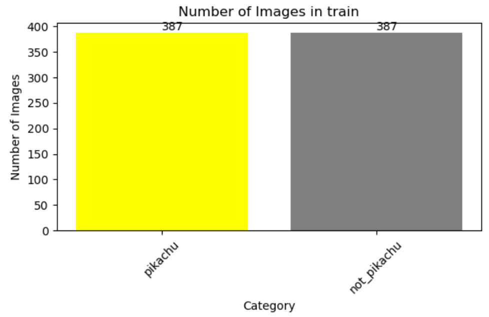
- **Specific Objectives**:
  - Data Preparation: Import, clean, and preprocess the image dataset to make it suitable for training the model.
  - Model Development: Design and implement a Convolutional Neural Network (CNN) for image classification.
  - Training and Validation: Train the model using the training dataset and validate its performance using the validation dataset.
  - Evaluation: Evaluate the model's performance on a test dataset using metrics such as accuracy, precision, recall, and F1-score.
  - Improvement: Implement transfer learning with pre-trained models (e.g., VGG16) to enhance model accuracy.

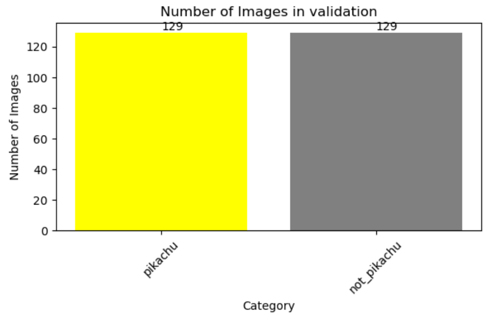
Data Description

- **Data Source:**
  - https://www.kaggle.com/datasets/hal0samuel/pikachu-classification-dataset

- **Data Overview**:
  - The dataset is a balanced binary classification dataset containing 645 images of Pikachu and 645 images of non-Pikachu. It includes separate subsets for training, testing, and validation.

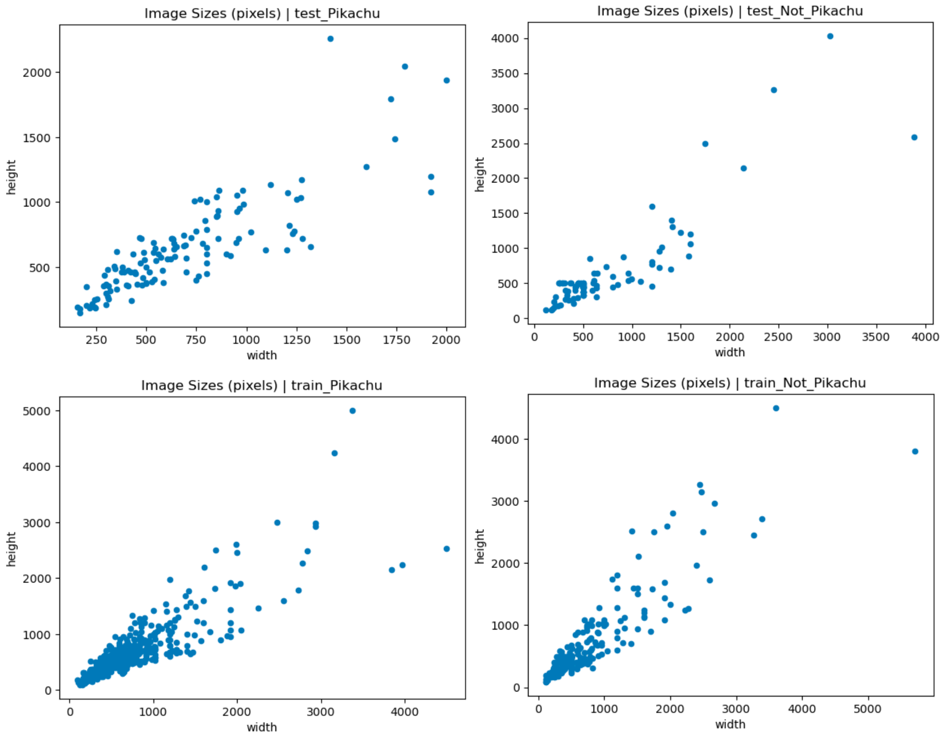






Data Description

➢ **Data Overview**:
- The scatter plots provide an overview of the image dimensions, highlighting the variability of the images.
- The dataset consists of images with varying dimensions, which necessitates preprocessing steps such as resizing to ensure uniformity for model training.



Data Modification

- **Subsets:**
  - Validation folders are merged into the training folder to streamline the dataset. The validation split will be performed dynamically during the image generation process.
- **Resizing**:
  - All images are resized to 120x120 pixels to ensure consistency and uniformity for model training.

Before Resizing

After Resizing

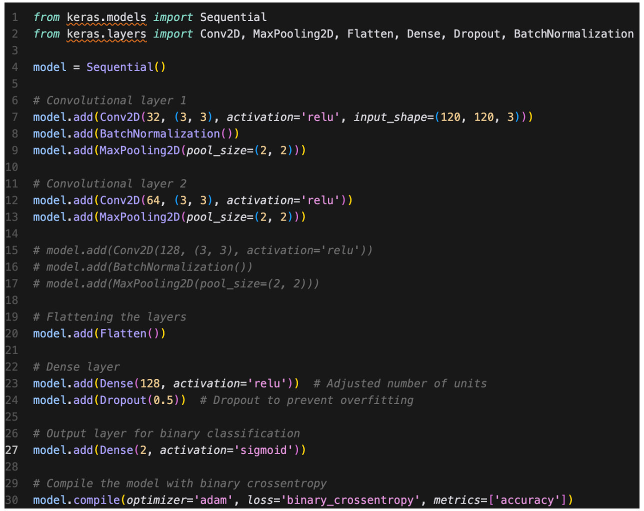
Data Processing

- **Image Data Generator:**
  - **Training data generator:**
    - Normalization: Rescale pixel values to the range [0, 1] by dividing by 255.
    - Data Augmentation: Apply various transformations to augment the dataset
    - Validation Split: Fraction of images reserved for validation(0.2)
  - **Validation data generator:**
    - Rescale pixel values to the range [0, 1] by dividing by 255.
    - Use 20% of the training data for validation.
  - **Test data generator:**
    - Rescale pixel values to the range [0, 1] by dividing by 255.
- **Summary:**
  - Training Set: 826 images belonging to 2 classes.
  - Validation Set: 206 images belonging to 2 classes.
  - Test Set: 258 images belonging to 2 classes.

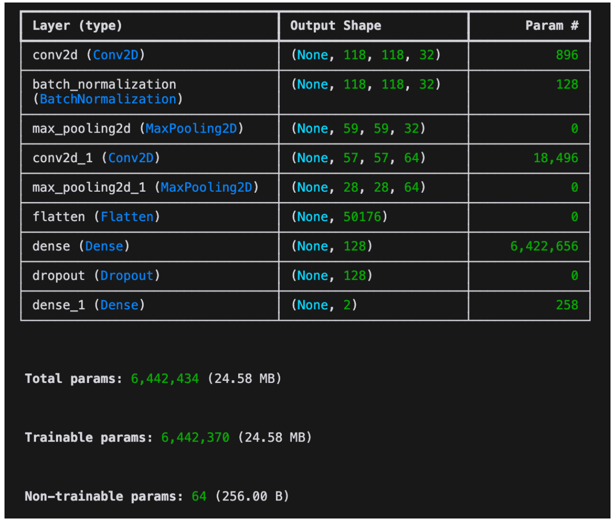
Model Architecture

➢ **Model Description:**

- The model used for classifying images into 'Pikachu' and 'Not Pikachu' categories is a Convolutional Neural Network (CNN).
- The architecture consists of several convolutional layers followed by pooling layers, and finally dense layers for classification.



```python
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization

model = Sequential()

# Convolutional layer 1
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(120, 120, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional layer 2
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# model.add(Conv2D(128, (3, 3), activation='relu'))
# model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
model.add(Flatten())

# Dense layer
model.add(Dense(128, activation='relu'))  # Adjusted number of units
model.add(Dropout(0.5))  # Dropout to prevent overfitting

# Output layer for binary classification
model.add(Dense(2, activation='sigmoid'))

# Compile the model with binary crossentropy
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```



```
Layer (type)                    Output Shape              Param #
conv2d (Conv2D)                 (None, 118, 118, 32)          896
batch_normalization             (None, 118, 118, 32)          128
(BatchNormalization)
max_pooling2d (MaxPooling2D)    (None, 59, 59, 32)              0
conv2d_1 (Conv2D)               (None, 57, 57, 64)         18,496
max_pooling2d_1 (MaxPooling2D)  (None, 28, 28, 64)              0
flatten (Flatten)               (None, 50176)                   0
dense (Dense)                   (None, 128)             6,422,656
dropout (Dropout)               (None, 128)                     0
dense_1 (Dense)                 (None, 2)                     258

Total params: 6,442,434 (24.58 MB)

Trainable params: 6,442,370 (24.58 MB)

Non-trainable params: 64 (256.00 B)
```

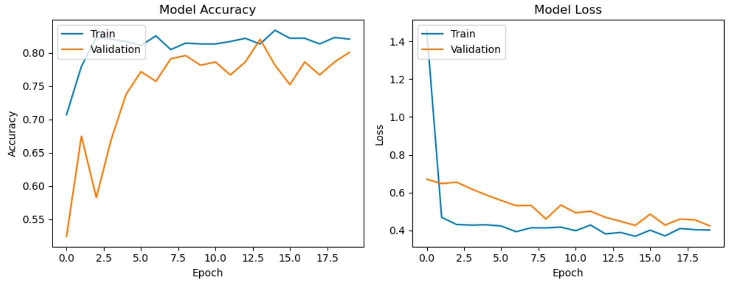
Model Architecture

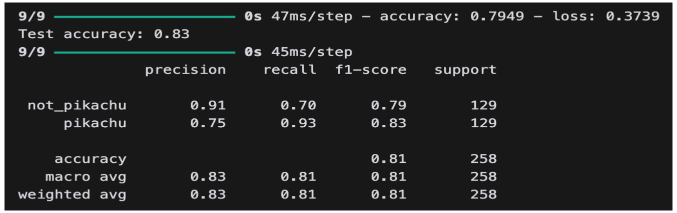
➢ **Model Training:**

- Dataset: The model is trained using the augmented and preprocessed dataset.
- Epochs: The model is trained for 20 epochs.
- Training and Validation: The training is performed using the train_generator, and validation is done using the validation_generator.
- Verbosity: Verbose mode is set to 1 for detailed logging during training.
- The model shows a consistent improvement in both accuracy and loss over the epochs.
- The final epoch indicates a training accuracy of 82.67% and a validation accuracy of 80.10%, demonstrating good generalization to the validation data.

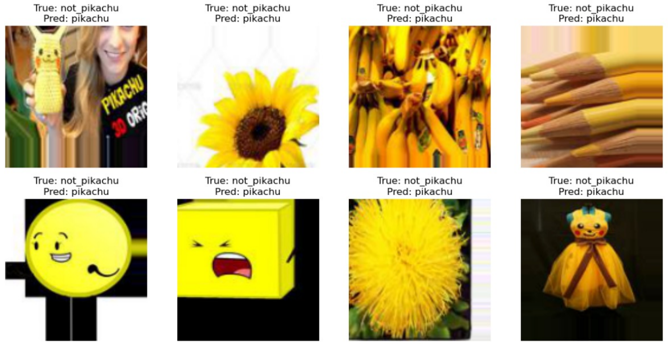
Model Evaluation

- **Accuracy and Loss Plots**
  - The model's accuracy increases steadily, reaching over 80% in both training and validation.
  - The loss decreases significantly in the first few epochs and then stab
- **Performance Metrics:**
  - Accuracy: The model achieved a test accuracy of 83%.
  - Loss: The final test loss was 0.3739.
- **Misclassification:**
  - Misclassified images (49 of them) displayed to provide insights into the types of errors the model makes.
  - Common misclassification involves images with similar Color and Texture, similar/complex Background and Context, Low Resolution or Blurriness.





```
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.7949 - loss: 0.3739
Test accuracy: 0.83
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
              precision    recall  f1-score   support

 not_pikachu       0.91      0.70      0.79       129
     pikachu       0.75      0.93      0.83       129

    accuracy                           0.81       258
   macro avg       0.83      0.81      0.81       258
weighted avg       0.83      0.81      0.81       258
```



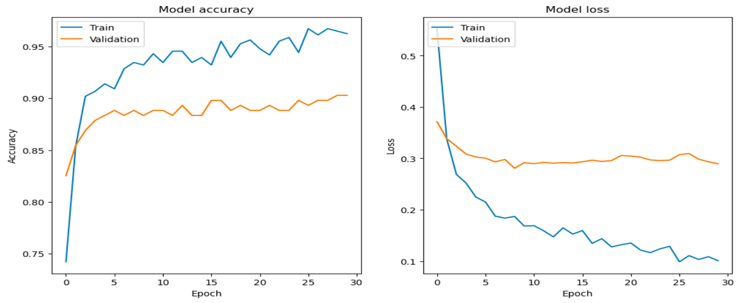
Transfer Learning with VGG16

- **Modeling with VGG16:**
  - Load the VGG16 model without the top layers and add custom dense layers to the VGG16 base model.
    “x = Flatten()(x) x = Dense(512, activation='relu')(x) x = Dropout(0.5)(x)”

- **Training the Model:**
  - Train the model for 30 epochs using the computed class weights to address class imbalance.
    “epochs=30, validation_data=validation_generator, class_weight=class_weights, verbose=1”
  - The final epoch indicates a training accuracy of 96.11% and a validation accuracy of 90.29%, demonstrating a better result then the first model.

Evaluating the New Model

- **Accuracy and Loss Plots**
  - The improved model shows smoother and more stable training and validation accuracy and loss curves, indicating better learning and generalization.
- **Performance Metrics:**
  - Accuracy: Accuracy increased from 83% to 95.02%.
  - Loss: The final test loss decreased from 0.3739 to 0.1388.
- **Misclassification:**
  - Fewer misclassifications of only 18 images, indicating enhanced robustness and improved ability to distinguish between similar-looking objects.



Metric	Initial Model	Improved Model
Accuracy	83%	95.02%
Loss	0.3739	0.1388
Precision		
- Not Pikachu	0.91	0.93
- Pikachu	0.75	0.96
Recall		
- Not Pikachu	0.70	0.96
- Pikachu	0.93	0.92
F1-score		
- Not Pikachu	0.79	0.94
- Pikachu	0.83	0.94
Macro Avg	0.81	0.94
Weighted Avg	0.81	0.94

Summary & Future Work:

- **Summary of the Project:**
  - **Objective**: The project aimed to build a machine learning model to classify images into 'Pikachu' and 'Not Pikachu' categories.
  - **Dataset**: Utilized a balanced dataset with 645 images of Pikachu and 645 images of non-Pikachu, divided into training, validation, and test sets.
  - **Initial Model**: Developed a Convolutional Neural Network (CNN) that achieved 83% accuracy with a loss of 0.3739.
  - **Improvements**: Applied transfer learning using the VGG16 model, which significantly enhanced the model’s performance.
  - **Final Model**: The improved model reached an accuracy of 95.02% and a loss of 0.1388, with high precision, recall, and F1-scores.
- **Future Work:**
  - **Further Tuning**: Explore hyperparameter tuning and other advanced models to push the accuracy even higher.
  - **Expanded Dataset**: Incorporate a larger and more diverse dataset to make the model more robust.
  - **Deployment**: Implement the model in a real-world application, such as an app for identifying Pikachu in images.

Conclusion:

This project demonstrates the power of machine learning and transfer learning in image classification tasks. The journey from building an initial CNN model to significantly improving it with VGG16 highlights the iterative nature of developing effective machine learning solutions.

For more details, code, and implementation steps, please visit the GitHub repository.