İZMİR
KÂTİP ÇELEBİ
ÜNİVERSİTESİ
2010
GRADUATE SCHOOL OF NATURAL
AND APPLIED SCIENCES

ROBOTİKTE MAKİNE GÖRMESİ

ROE510

Homework 6

Onuralp Arslan

2024

Step1: Data Gathering

Data required for this application is collected from a kaggle dataset. Dataset named Dogs vs. Cats is supplied by kaggle for Playground prediction competition and can be found at https://www.kaggle.com/c/dogs-vs-cats/data .

Step2: Preparing Development

Training will be performed on Colab, to quickly transfer the training data I have shared on my github with credits to the original data source and use “wget” command to quickly load to my workspace.

Github link to ipynb : https://github.com/onuralpArsln/MlAiTutorialProjects/blob/main/5-ImageClass/CatDog/modelTrain.ipynb

Command to quickly load data:

!wget https://github.com/onuralpArsln/MlAiTutorialProjects/raw/main/5-ImageClass/CatDog/data.zip -O data.zip

After using “!unzip -q data.zip” data is unzipped.

Step3 : Data Preprocessing

Images are scaled and then divided into train and validation sets. This allows uniform inputs to network and ability to test success of training.

Step 4: Model Building

A simple model with requested layers is built.

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

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

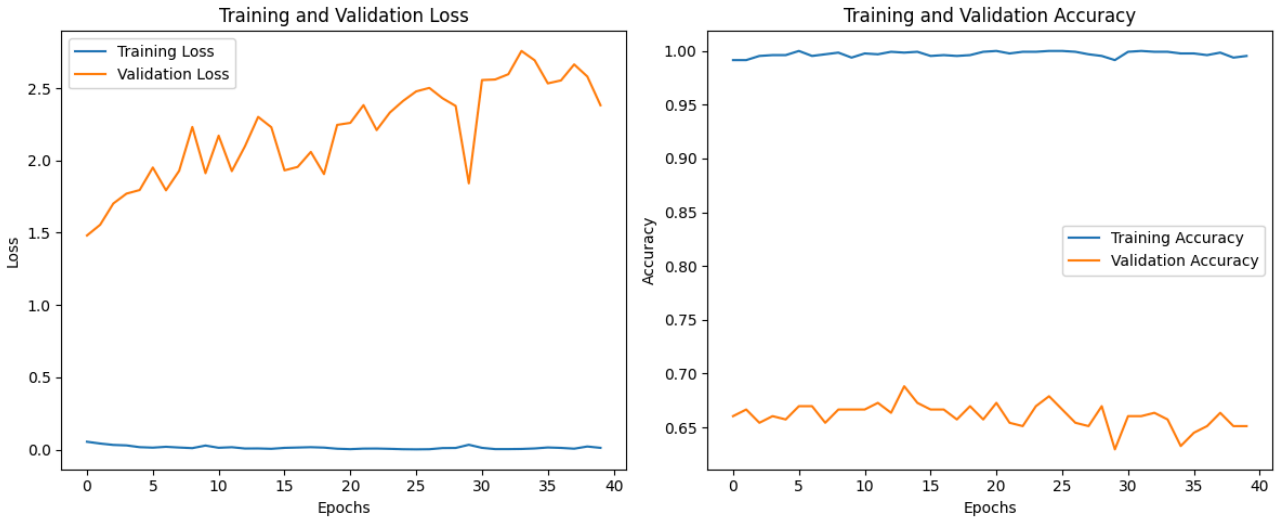
Step 5: Training

Using model.fit training is performed. I have started with batch size 16 and epoch size 10 which was inadequate for the given task. I have increased the batch size and epochs. This given state of hyperparameters of 40 epoch and 32 batch size are the fastest trained values that are still able to perform at acceptable success.

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=40,
    batch_size=32
)
```

Step 6: Evaluation

Randomly selected five visuals to test results with only one is miss classed.



In the given graphic we can see training loss is approaching zero while validation loss is increasing, this might indicate an overfitting situation.

Step 7: New Model and Evaluation

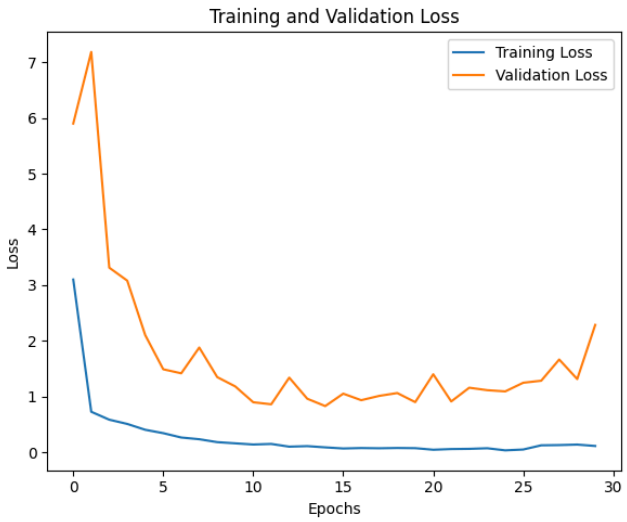
New model is built with fewer epochs and obtained a better result at early steps.

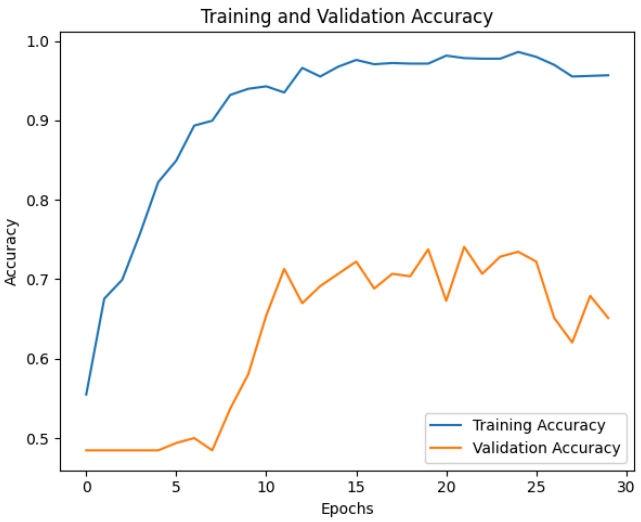
```
model = Sequential([
    # 1. Convolution + MaxPooling
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # 2. Convolution + MaxPooling
    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Convolution + MaxPooling
    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Fully Connected Layers
    Flatten(),
    Dense(256, activation='relu'),  # Daha büyük bir dense layer ekledik
    Dropout(0.5),                   # Regularization için dropout ekledik
    Dense(1, activation='sigmoid')  # Binary classification için
])
```





In these results we can see that validation loss decreases with training loss at early epoch showing that overfitting is reduced initially. Yet there is still room for improvement. More data and better model structure can solve this problem.