PROJECT ASSIGNMENT

Dogs vs. Cats Image Classification using Convolutional Neural Networks

ISE741

DEEP LEARNING

by

VASHISTA A N (1MS20IS131)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

August 2023

Problem Statement:

Cats vs Dogs:

The cats vs dogs is a good project to start as a beginner in deep learning. You can build a model that takes an image as input and determines whether the image contains a picture of a dog or a cat.

Abstract:

In the realm of computer vision, image classification stands as a cornerstone task. With the proliferation of deep learning techniques, particularly Convolutional Neural Networks (CNNs), the accuracy and efficiency of image classification have witnessed significant advancements. This project delves into harnessing the capabilities of CNNs to distinguish between images of dogs and cats.

Utilizing the Dogs vs. Cats dataset sourced from Kaggle, which comprises 25,000 labeled images, a deep learning model was architected and trained to classify these images. The dataset was systematically partitioned into training (80%) and validation (20%) sets. The designed CNN model incorporated multiple convolutional layers paired with max-pooling layers, followed by dense layers to perform the classification.

Data augmentation techniques were employed during training to enhance the model's generalization capabilities. The model was optimized using the Adam optimizer and was trained with a binary cross-entropy loss function.

Post-training, the model's performance was evaluated on the validation set, yielding promising results in terms of accuracy and other metrics. The project not only underscores the efficacy of CNNs in image classification tasks but also provides a foundational framework for further exploration into more complex image classification challenges.

Objective:

To design and implement a deep learning model that can accurately classify images as either a dog or a cat.

Introduction and concept:

Image classification is a fundamental task in computer vision. With the advent of deep learning and especially Convolutional Neural Networks (CNNs), the accuracy of image classification tasks has seen significant improvement. This project aims to harness the power of CNNs to classify images of dogs and cats.

The project "Cats vs Dogs" is an introductory venture into the world of deep learning, specifically focusing on image classification within the domain of computer vision. The primary objective is to design and implement a deep learning model capable of accurately classifying images as either a dog or a cat. This task is not just a playful exercise but serves as a foundational framework for understanding more complex image classification challenges that are prevalent in various sectors like healthcare, security, and automation.

The project leverages the power of Convolutional Neural Networks (CNNs), a class of deep neural networks most commonly applied to analysing visual imagery. CNNs have been revolutionary in the field of computer vision, offering high accuracy in tasks ranging from object detection to semantic segmentation. The project aims to demonstrate the efficacy of CNNs in classifying images of dogs and cats, thereby providing a stepping stone for those interested in diving deeper into the realm of computer vision and deep learning.

Dataset and description:

The dataset used for this project is the "Dogs vs. Cats" dataset available on Kaggle. It consists of 25,000 labelled images, with each image explicitly labeled as either a dog or a cat. The images come in various resolutions and aspect ratios, making the dataset diverse and more challenging for the model.

The dataset is divided into two main sets:

1. **Training Set**: Comprises 80% of the total dataset (20,000 images). This set is used to train the model.
2. **Validation Set**: Comprises the remaining 20% (5,000 images). This set is used to validate the model's performance.

Each of these sets is further divided into two subdirectories, one for each class (dogs and cats), making it easier for the model to fetch and learn from the data.

To make the model more robust and generalized, data augmentation techniques are applied to the training set. These techniques include:

- Random Rotations
- Horizontal and Vertical Flips
- Zooming

Model Architecture:

The architecture of the model is designed using TensorFlow's Keras API, a high-level neural networks API running on top of TensorFlow, which makes it easier to construct and train complex models.

The model consists of the following layers:

1. **Input Layer**: Takes in an image of a predefined shape.
2. **Convolutional Layers (Conv2D)**: These are the core building blocks of a CNN. The Conv2D layers are used to create a feature map that represents the input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. We use more than one convolutional layer to improve the model's performance.
3. **Max-Pooling Layers (MaxPooling2D)**: These layers are used to reduce the dimensions of the feature maps, thereby reducing the number of parameters and computation in the network. This also helps in making the detection of features invariant to scale and orientation changes.
4. **Flatten Layer**: This layer flattens the 2D matrix data to a vector so that it can be fed into the dense layers.
5. **Dense Layers**: These are densely connected neural network layers where each input node is connected to each output node. The first dense layer has more neurons, followed by a second dense layer that has one neuron to output either a dog or a cat.
6. **Activation Functions**: ReLU (Rectified Linear Unit) activation functions are used in the convolutional and dense layers to introduce non-linearity into the network. The output layer uses a sigmoid activation function to output probabilities.
7. **Optimizer and Loss Function**: The Adam optimizer is used for optimizing the model, and the binary cross-entropy loss function is used as it's a binary classification problem.
8.

Data Augmentation and Training

Data augmentation techniques like rotation, flipping, and zooming are employed to artificially increase the size of the training dataset. This helps in enhancing the model's ability to generalize well to new, unseen data. The model is trained on 80% of the data, and its performance is validated on the remaining 20%.

```python
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(units=128, activation='relu'))
    model.add(Dense(units=1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```
✓ 0.0s

1. **Model Design/Definition**:
    a. The model is a Convolutional Neural Network (CNN) built using TensorFlow's Keras API.
    b. The model consists of two sets of Conv2D and MaxPooling2D layers, followed by a Flatten layer and two Dense layers.
2. **Training**:
    a. Data Augmentation is used to artificially increase the size of the training dataset.
    b. The model is trained using the Adam optimizer and binary cross-entropy loss.
3. **Evaluation:** A function is implemented to classify a user-provided image.

```python
def train_model(model, train_dir, validation_dir):
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

    test_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

    validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

    model.fit(
        train_generator,
        steps_per_epoch=len(train_generator),
        epochs=25,
        validation_data=validation_generator,
        validation_steps=len(validation_generator))
```
✓ 0.0s

```
Found 20000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
Epoch 1/25
625/625 [==============================] - 261s 415ms/step - loss: 0.6196 - accuracy: 0.6518 - val_loss: 0.5629 - val_accuracy: 0.6980
Epoch 2/25
625/625 [==============================] - 105s 167ms/step - loss: 0.5371 - accuracy: 0.7332 - val_loss: 0.5425 - val_accuracy: 0.7324
Epoch 3/25
625/625 [==============================] - 103s 164ms/step - loss: 0.4987 - accuracy: 0.7511 - val_loss: 0.4564 - val_accuracy: 0.7804
Epoch 4/25
625/625 [==============================] - 109s 174ms/step - loss: 0.4670 - accuracy: 0.7742 - val_loss: 0.5007 - val_accuracy: 0.7546
Epoch 5/25
625/625 [==============================] - 103s 165ms/step - loss: 0.4455 - accuracy: 0.7888 - val_loss: 0.4789 - val_accuracy: 0.7798
Epoch 6/25
625/625 [==============================] - 100s 160ms/step - loss: 0.4279 - accuracy: 0.7991 - val_loss: 0.4200 - val_accuracy: 0.8018
Epoch 7/25
625/625 [==============================] - 109s 175ms/step - loss: 0.4103 - accuracy: 0.8134 - val_loss: 0.4459 - val_accuracy: 0.7916
Epoch 8/25
625/625 [==============================] - 103s 164ms/step - loss: 0.3941 - accuracy: 0.8205 - val_loss: 0.4016 - val_accuracy: 0.8212
Epoch 9/25
625/625 [==============================] - 102s 163ms/step - loss: 0.3840 - accuracy: 0.8265 - val_loss: 0.4361 - val_accuracy: 0.7994
Epoch 10/25
625/625 [==============================] - 104s 166ms/step - loss: 0.3670 - accuracy: 0.8325 - val_loss: 0.4621 - val_accuracy: 0.7916
Epoch 11/25
625/625 [==============================] - 103s 164ms/step - loss: 0.3519 - accuracy: 0.8428 - val_loss: 0.4178 - val_accuracy: 0.8166
Epoch 12/25
...
Epoch 24/25
625/625 [==============================] - 115s 184ms/step - loss: 0.2045 - accuracy: 0.9165 - val_loss: 0.4641 - val_accuracy: 0.8344
Epoch 25/25
625/625 [==============================] - 109s 174ms/step - loss: 0.2045 - accuracy: 0.9144 - val_loss: 0.5006 - val_accuracy: 0.8270
```

Process Flow:

1. **Data Collection**: Obtain the Dogs vs. Cats dataset from Kaggle.
2. **Data Preparation**: Organize the dataset into training and validation sets.
3. **Model Definition**: Define the architecture of the CNN.
4. **Model Training**: Train the model using the training set.
5. **Model Evaluation**: Evaluate the model's performance on the validation set.
6. **User Input Classification**: Classify user-provided images.

Results:

The model's performance can be evaluated based on its accuracy on the validation set. For example, after training, the model achieved an accuracy of 90% on the validation set (this is a placeholder; you should replace it with your actual result). The accuracy metric indicates the proportion of correctly classified images out of the total images in the validation set.

```
Found 5000 images belonging to 2 classes.
157/157 [==============================] - 11s 65ms/step
Confusion Matrix:
[[1945  555]
 [ 310 2190]]

Classification Report:
              precision    recall  f1-score   support

         cat       0.86      0.78      0.82      2500
         dog       0.80      0.88      0.84      2500

    accuracy                           0.83      5000
   macro avg       0.83      0.83      0.83      5000
weighted avg       0.83      0.83      0.83      5000
```

```
image_path = 'test1/test1/12471.jpg'
print(f"The image is of a {classify_image(model, image_path)}")
```

✓ 0.2s

```
1/1 [==============================] - 0s 108ms/step
The image is of a dog
```

Conclusion:

The project demonstrates the effectiveness of Convolutional Neural Networks in image classification tasks. With appropriate data augmentation, optimization, and model architecture, high accuracy can be achieved in distinguishing between images of dogs and cats.