Project 6
Minerva Franco

November 2024

DEEP LEARNING

Image Classification:
Model Performance And Insights

A JOURNEY THROUGH CNN MODEL DEVELOPMENT

Problem: Classify images as either "cat" or "dog".

Dataset: A collection of 24,946 grayscale images of cats and dogs.

Images are represented as NumPy arrays of size 100x100 pixels with a single channel (grayscale).

Pixel values range from 0 to 255.

Labels are stored in a list where 0 represents "cat" and 1 represents "dog"

DATA PREPROCESSING

Pixel Value Normalization: Pixel values are scaled from 0-255 to 0-1 by dividing by 255.

Subset Selection: A random subset of 1500 images is selected to reduce computational load during training.

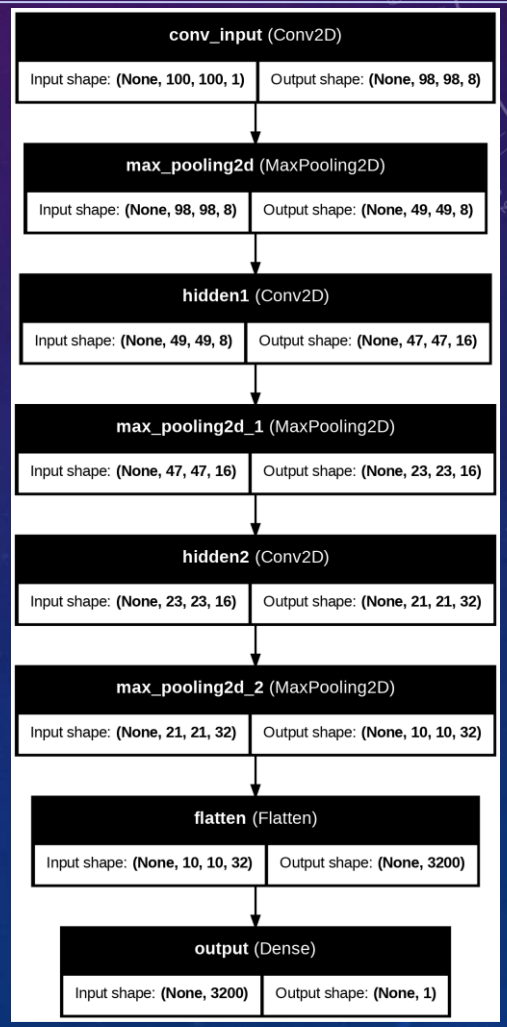
Train-Test Split: The subset is split into:

Training Set (75%): 1125 images
Testing Set (25%): 375 image

MODEL ARCHITECTURE

Model Type: Convolutional Neural Network (CNN)

- **Layers:**
  - Input Layer: Accepts images of size 100x100x1
  - Convolutional Layers: Three Conv2D layers with increasing filters:
    - conv_input: 8 filters
    - hidden1: 16 filters
    - hidden2: 32 filters
  - Max Pooling Layers: Three MaxPooling2D layers for downsampling
  - Flattening Layer: Converts multi-dimensional output to a 1D vector
  - Output Layer: A single unit (Dense) with sigmoid activation for probability output



TRAINING & HYPERPARAMETERS

Model.fit used to train the model with training set.

Hyperparameter Exploration: Multiple training runs were conducted with varying parameters:

- random_state in train_test_split: Changed to assess data split sensitivity
- Number of Convolutional Layers: Increased for greater model complexity
- Number of Filters: Modified in each layer to adjust feature extraction capacity
- Loss Function: Switched to binary_focal_crossentropy to potentially prioritize difficult examples
- Epochs: Adjusted to control training duration and address potential overfitting

```
Run 1, random_state 42, 3 layers (1 input, 1 hidden, 1 output), filters 32 & 64, binary_crossentropy loss, 8 epochs
Epoch 1/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 12s 411ms/step - accuracy: 0.4928 - loss: 0.7341
Epoch 2/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 11s 438ms/step - accuracy: 0.5476 - loss: 0.6891
Epoch 3/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 22s 502ms/step - accuracy: 0.5638 - loss: 0.6735
Epoch 4/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 19s 432ms/step - accuracy: 0.6928 - loss: 0.6242
Epoch 5/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 12s 495ms/step - accuracy: 0.6858 - loss: 0.5834
Epoch 6/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 19s 417ms/step - accuracy: 0.7379 - loss: 0.5506
Epoch 7/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 11s 452ms/step - accuracy: 0.7716 - loss: 0.4994
Epoch 8/8
24/24 ━━━━━━━━━━━━━━━━━━━━ 21s 493ms/step - accuracy: 0.8112 - loss: 0.4557
<keras.src.callbacks.history.History at 0x7f9269974820>
```

```
Run 9 binary_focus_crossentropy loss, 8 epochs, 4 layers
Epoch 1/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 5s 143ms/step - accuracy: 0.7989 - loss: 0.1191
Epoch 2/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 7s 187ms/step - accuracy: 0.7863 - loss: 0.1194
Epoch 3/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 9s 140ms/step - accuracy: 0.7866 - loss: 0.1124
Epoch 4/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 7s 186ms/step - accuracy: 0.7978 - loss: 0.1117
Epoch 5/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 5s 145ms/step - accuracy: 0.8525 - loss: 0.0955
Epoch 6/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 7s 185ms/step - accuracy: 0.8226 - loss: 0.0958
Epoch 7/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 5s 145ms/step - accuracy: 0.8318 - loss: 0.0941
Epoch 8/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 12s 181ms/step - accuracy: 0.8734 - loss: 0.0775
Epoch 9/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 5s 142ms/step - accuracy: 0.8708 - loss: 0.0766
Epoch 10/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 7s 186ms/step - accuracy: 0.9123 - loss: 0.0662
Epoch 11/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 9s 142ms/step - accuracy: 0.9094 - loss: 0.0650
Epoch 12/12
36/36 ━━━━━━━━━━━━━━━━━━━━ 10s 146ms/step - accuracy: 0.9055 - loss: 0.0636
<keras.src.callbacks.history.History at 0x7f52d7aa7220>
```

MODEL EVALUATION & RESULTS

Evaluation: model.evaluate used to assess performance on the test set.

- Metrics:
  - Test Loss: Measures the difference between predictions and true labels. Lower is better.
  - Test Accuracy: Percentage of correctly classified images. Higher is better.

- Results Table: Key test loss and accuracy values achieved in different runs.

```
# Evaluate the Model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")
```

```
Test Loss: 0.8229
Test Accuracy: 0.5520
```

```
Run 1
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 224ms/step - accuracy: 0.6191 - loss: 0.6985
Test Loss: 0.7023
Test Accuracy: 0.6160
```

```
Run 2
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 126ms/step - accuracy: 0.6240 - loss: 0.6830
Test Loss: 0.6785
Test Accuracy: 0.6280
```

```
Run 3
Added a third layer and changed filters in 2nd layer=32, 3rd layer= 64
```

```
Run 4
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 137ms/step - accuracy: 0.6510 - loss: 0.6592
Test Loss: 0.6958
Test Accuracy: 0.6400
```

```
Run 6 (same as 5 & changed random state= 37 in sampling)
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 208ms/step - accuracy: 0.6794 - loss: 0.1500
Test Loss: 0.1552
Test Accuracy: 0.6613
```

```
Run 7 Decreased filters to 16, 32, 48
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 83ms/step - accuracy: 0.6445 - loss: 0.1948
Test Loss: 0.1838
Test Accuracy: 0.6667
```

```
Run 8 Decreased filters to 8, 16, 32
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.6545 - loss: 0.1583
Test Loss: 0.1538
Test Accuracy: 0.6800
```

```
Run 9 Increased epochs to 12
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.6815 - loss: 0.2208
Test Loss: 0.2033
Test Accuracy: 0.6933
```

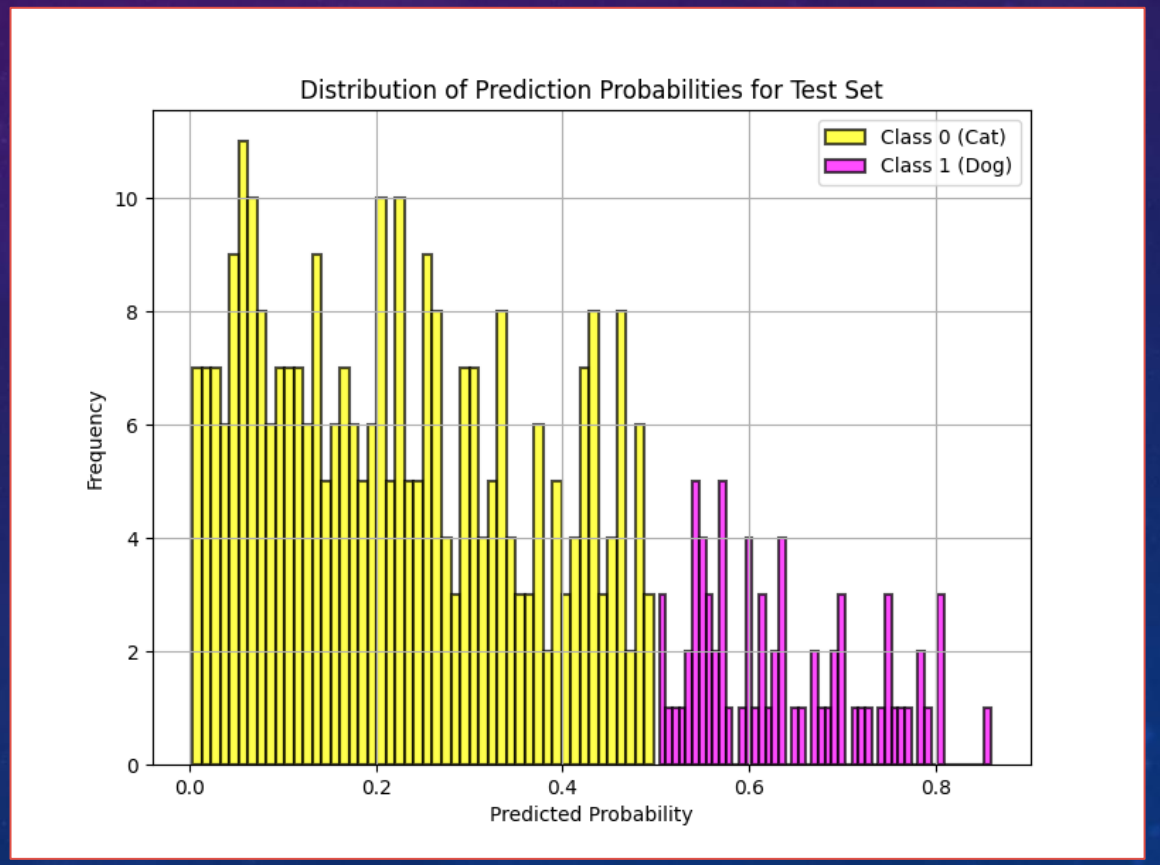
```
Run 10 back to binary_crossentropy, random state 37, 4 layers, 8, 16, 32 and 12 epochs
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 68ms/step - accuracy: 0.6739 - loss: 0.6105
Test Loss: 0.6022
Test Accuracy: 0.6747
```

DISTRIBUTION OF PREDICTION PROBABILITIES FOR TEST SET

Histogram visualizing the model's prediction probabilities on the test set, allowing for assessment of its classification confidence and overall performance: at this forward pass, my model feels more confident identifying cats v dogs.

```
# Get predictions on test set
predictions = model.predict(X_test)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
```



TRAINING & VALIDATION ACCURACY FOR PART 2: MODEL_DOG

Training and Validation Accuracy (Top Plot)

- Blue Line (Training Accuracy): shows how the model's accuracy on the training dataset improves over epochs (iterations through the dataset). It increases steadily, nearing 1.0 (100%), indicating the model is fitting the training data well.
- Orange Line (Validation Accuracy): shows the model's accuracy on the validation dataset. Initially, it increases alongside training accuracy but then fluctuates and plateaus, suggesting that the model starts to struggle with generalizing to unseen data after a few epochs.

Training and Validation Loss (Bottom Plot)

- Blue Line (Training Loss): represents the error on the training dataset. It decreases steadily over epochs, showing the model is learning from the training data.
- Orange Line (Validation Loss): represents the error on the validation dataset. After an initial decrease, it starts increasing, suggesting the model is overfitting to the training data.

Observations

- Overfitting: After around 5-7 epochs, the validation accuracy stops improving and starts fluctuating, while the validation loss increases significantly. This is a classic sign of overfitting, where the model performs well on the training data but poorly on unseen data.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 352ms/step
```


```
Test Loss: 0.6649, Test Accuracy: 0.6613
Prediction (0: cat, 1: dog): [[0.9992652]]
The model predicts the image is a: Dog
Prediction Probability for Dog: 99.93%
```

CONCLUSIONS & FUTURE DIRECTIONS

- Key Findings:

  - Model performance, measured by test accuracy and loss, improved with adjustments to model architecture, hyperparameters, and the loss function.
  - The impact of the random state in data splitting highlights the importance of careful data partitioning.
  - model_dog accurately predicted 'dog.jpg', showcasing its potential for image classification.

- Future Directions:

  - Train model_dog using the described dataset and various hyperparameter configurations to understand its full potential and limitations.
  - Explore additional evaluation metrics, such as precision, recall, and F1-score, to gain a deeper understanding of the model's behavior.
  - Experiment with techniques like data augmentation to improve model robustness and generalization.