FINAL PROJECT

Student's name: THUY THIEN TRAM NGUYEN

Student's ID: 811293917

In this final project, my chosen topic is the application of CNN (Convolutional Neural Network) in Computer Vision. To demonstrate its effectiveness and support my conclusion, I decided to use a CNN model to classify whether a patient has pneumonia or is healthy. The data was sourced from Kaggle under the name Chest X-Ray Images (Pneumonia), consisting of 5,863 X-ray images of lungs, divided into two categories: normal and pneumonia.

METHOD:

- First of all, creates a smaller subset of a large dataset for a NORMAL AND PNEUMONIA classification task by randomly selecting 500 images each for normal and pneumonia from the training set, and 8 pictures from validation and 50 from testset.

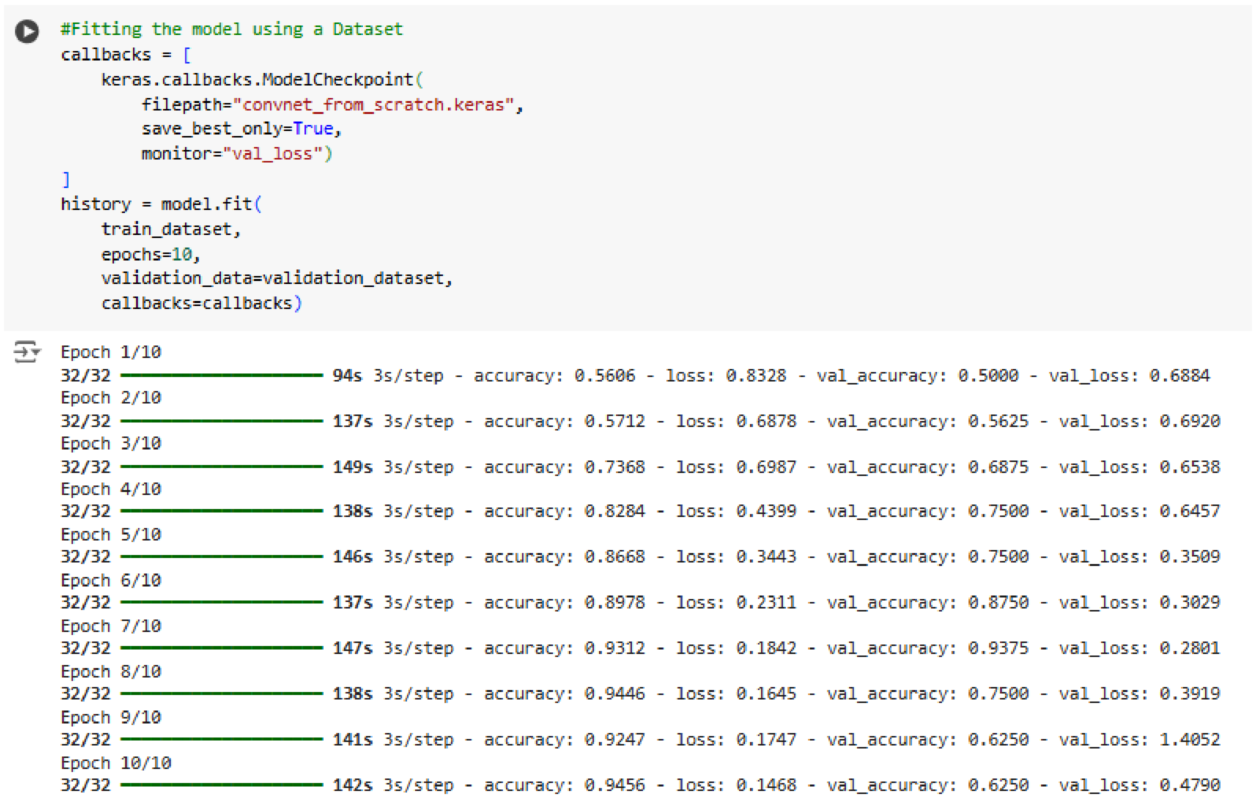
MODEL'S PARAMETERS:

- Input layer: keras.Input with shape (180, 180, 3) for RGB images.
- Normalization layer: Rescaling(1./255), which scales pixel values between 0 and 1.
- 5 Convolutional layers (Conv2D) with ReLU activation, organized as follows:
- ▪ 1st: 32 filters, 3x3 kernel
- ▪ 2nd: 64 filters, 3x3 kernel
- ▪ 3rd: 128 filters, 3x3 kernel
- ▪ 4th: 256 filters, 3x3 kernel
- ▪ 5th: 256 filters, 3x3 kernel
- 5 Pooling layers (MaxPooling2D), each with a 2x2 pool size, following each Conv2D layer.
- Flatten layer: Converts 2D features to a 1D vector.
- Output layer: Dense layer with 1 unit and a sigmoid activation for binary classification.
- Layers.RandomFlip("horizontal")
- Layers.RandomRotation(0.1)
- Layers.RandomZoom(0.2)
- Dropout value =0.5

➔ In this scenario, I started with the filters increase from 32 to 256 across layers to progressively capture more complex patterns. Kernel 3x3 is effectively captures details, it allows the model to learn from a part with size 3x3 pixel of the picture to it could mainly focus on details. Each pooling layer with a 2x2 size progressively reduces spatial dimensions by half, simplifying the data and focusing on prominent features while controlling overfitting and minimizing computational load. Dropout with a rate of 0.5 means that 50% of the neurons in that layer will be randomly deactivated during each training iteration

MODEL'S PERFORMANCE:

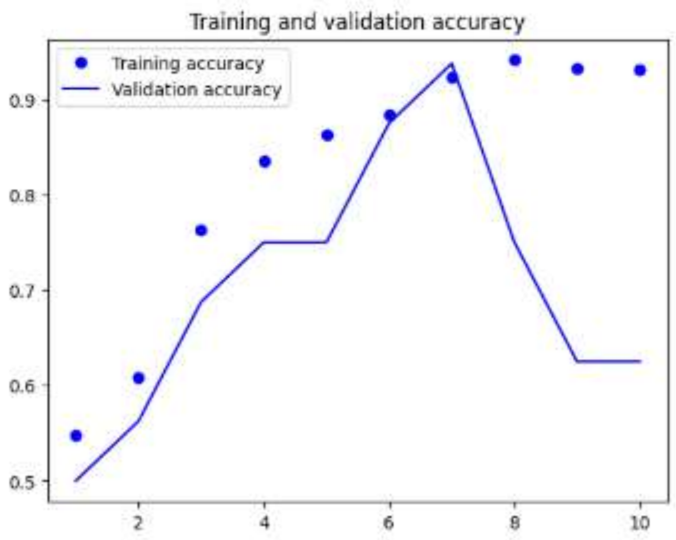
A) Accuracy in Training and Validation dataset

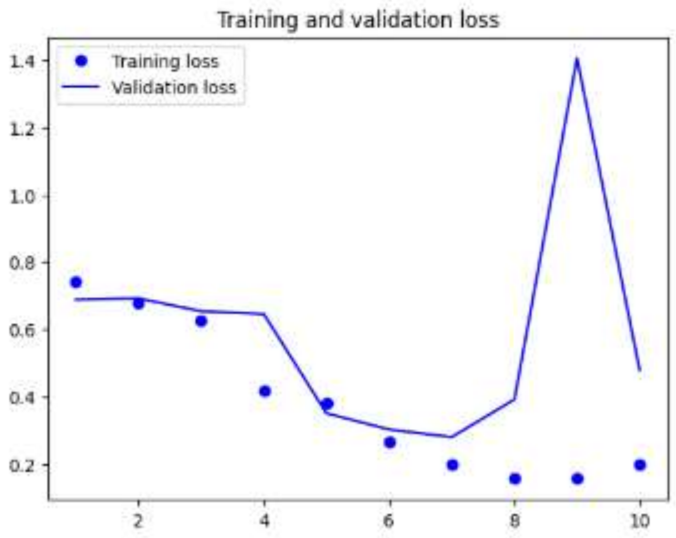


```
#Fitting the model using a Dataset
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 94s 3s/step - accuracy: 0.5606 - loss: 0.8328 - val_accuracy: 0.5000 - val_loss: 0.6884
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 137s 3s/step - accuracy: 0.5712 - loss: 0.6878 - val_accuracy: 0.5625 - val_loss: 0.6920
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 149s 3s/step - accuracy: 0.7368 - loss: 0.6987 - val_accuracy: 0.6875 - val_loss: 0.6538
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 138s 3s/step - accuracy: 0.8284 - loss: 0.4399 - val_accuracy: 0.7500 - val_loss: 0.6457
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 146s 3s/step - accuracy: 0.8668 - loss: 0.3443 - val_accuracy: 0.7500 - val_loss: 0.3509
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 137s 3s/step - accuracy: 0.8978 - loss: 0.2311 - val_accuracy: 0.8750 - val_loss: 0.3029
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 147s 3s/step - accuracy: 0.9312 - loss: 0.1842 - val_accuracy: 0.9375 - val_loss: 0.2801
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 138s 3s/step - accuracy: 0.9446 - loss: 0.1645 - val_accuracy: 0.7500 - val_loss: 0.3919
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 141s 3s/step - accuracy: 0.9247 - loss: 0.1747 - val_accuracy: 0.6250 - val_loss: 1.4052
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 142s 3s/step - accuracy: 0.9456 - loss: 0.1468 - val_accuracy: 0.6250 - val_loss: 0.4790
```

- Generally, the algorithm shows their well study based on the Training accuarcy increased from 56% up to 86% in epoch 5, the accuracy value in validation set also dramtically improved from 50% (Epoch 1) to 75% (Epoch 5). The improvement in both dataset continously increased until the accuracy of Validation dropped down from 93% to 75% in Epoch 8. It can be predicted that the model might capture too much details and become **Overfitting.**



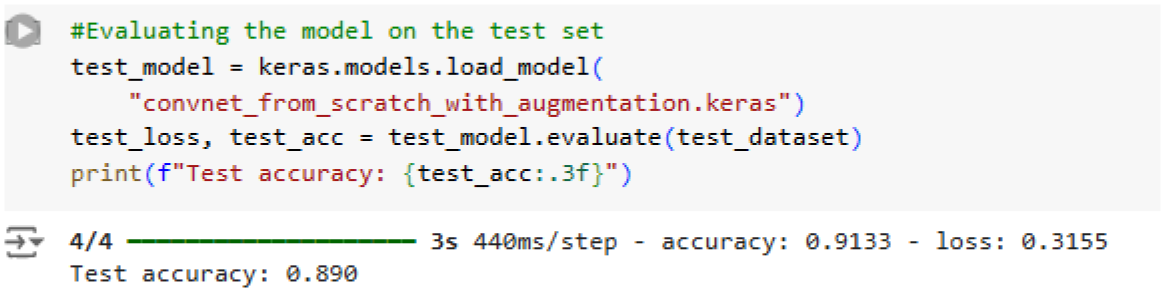


B) Accuracy in Testset:

```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 5s 928ms/step - accuracy: 0.7924 - loss: 0.4191
Test accuracy: 0.770
```

After that, to improve accuracy, I increased the data samples using data augmentation methods, including horizontal flipping, rotating by 0.1 degrees, and zooming by 0.2. I then retrained the data through the same Convolutional layers as before. The final prediction accuracy improved from 77% to 89%.



```python
#Evaluating the model on the test set
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 3s 440ms/step - accuracy: 0.9133 - loss: 0.3155
Test accuracy: 0.890
```

CONCLUSION:

This application demonstrates the effectiveness of CNN in predicting pneumonia in patients, achieving an accuracy of 89%. However, the results need to be validated to ensure that the model has truly learned well and provides reliable outcomes. Additionally, due to limitations in the dataset, the diversity is not fully represented, and the model is not equipped to address more complex questions such as: What is the severity level of pneumonia? Is it mild or severe? Or tracking the development of cancer cells to predict whether a tumor is malignant or benign? These are applications where CNN can further support Computer Vision. However, addressing such requirements would involve more complexity and is typically carried out by large organizations. This presentation is intended for reference purposes only.