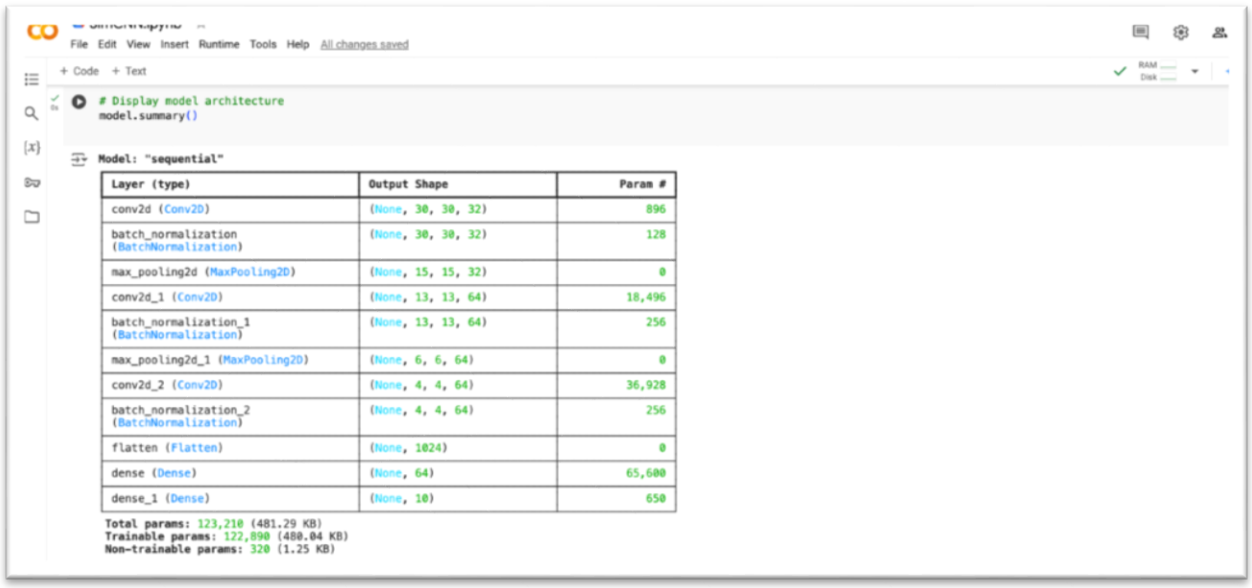
DEEP LEARNING PROJECT ASSIGNMENT

Sri Sakticharan Nirmal Kumar
1337576

SIMCNN : filename :simCNN2.py

Using the training and testing data from CIFAR-10, perform classification with SimCNN and find the classification accuracy of SimCNN with parameters in table 1

Code:

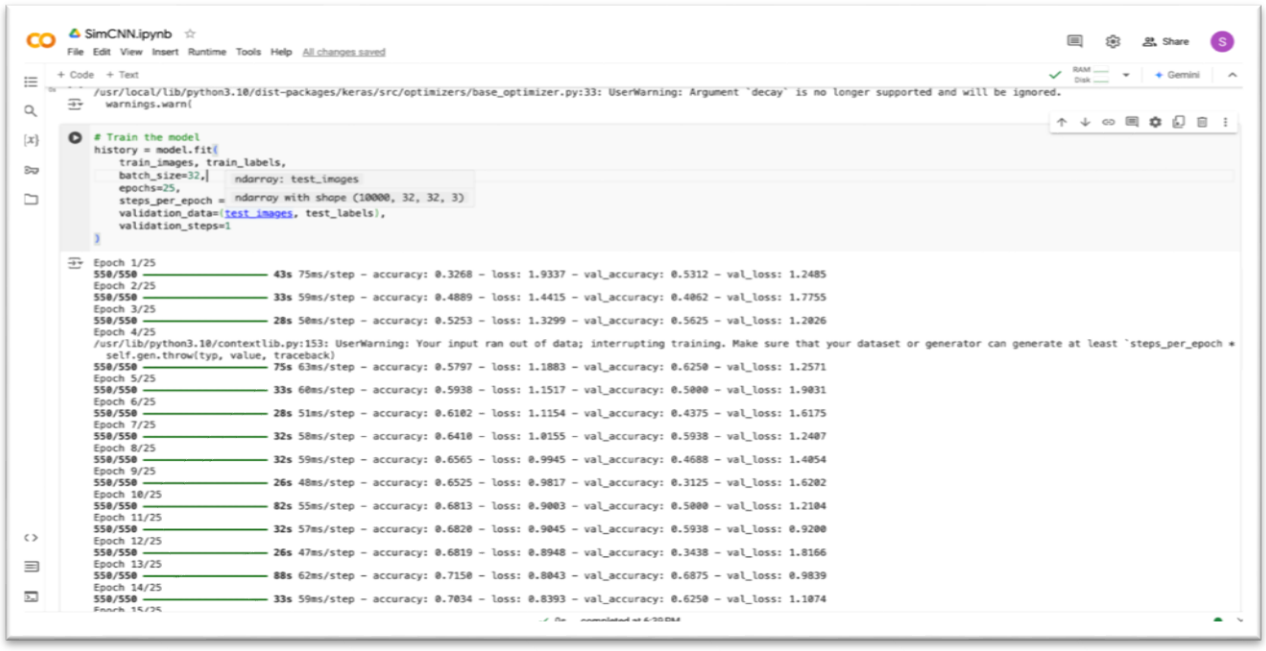


```
# Display model architecture
model.summary()
```

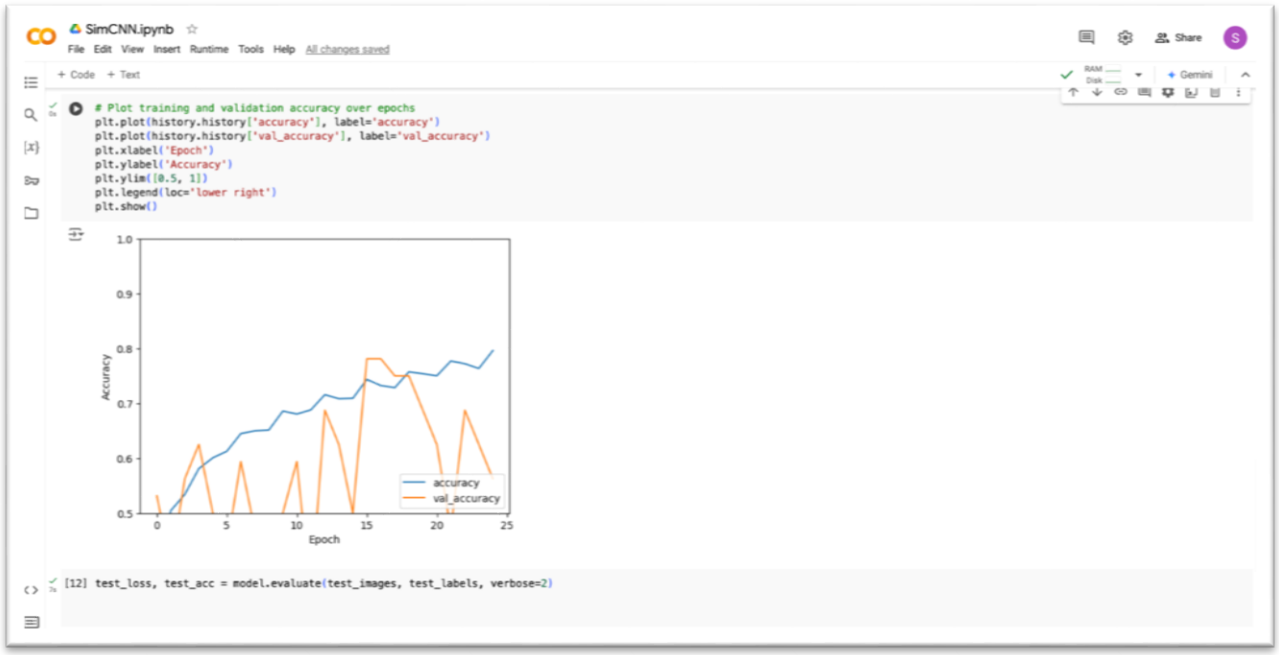
Model: "sequential"

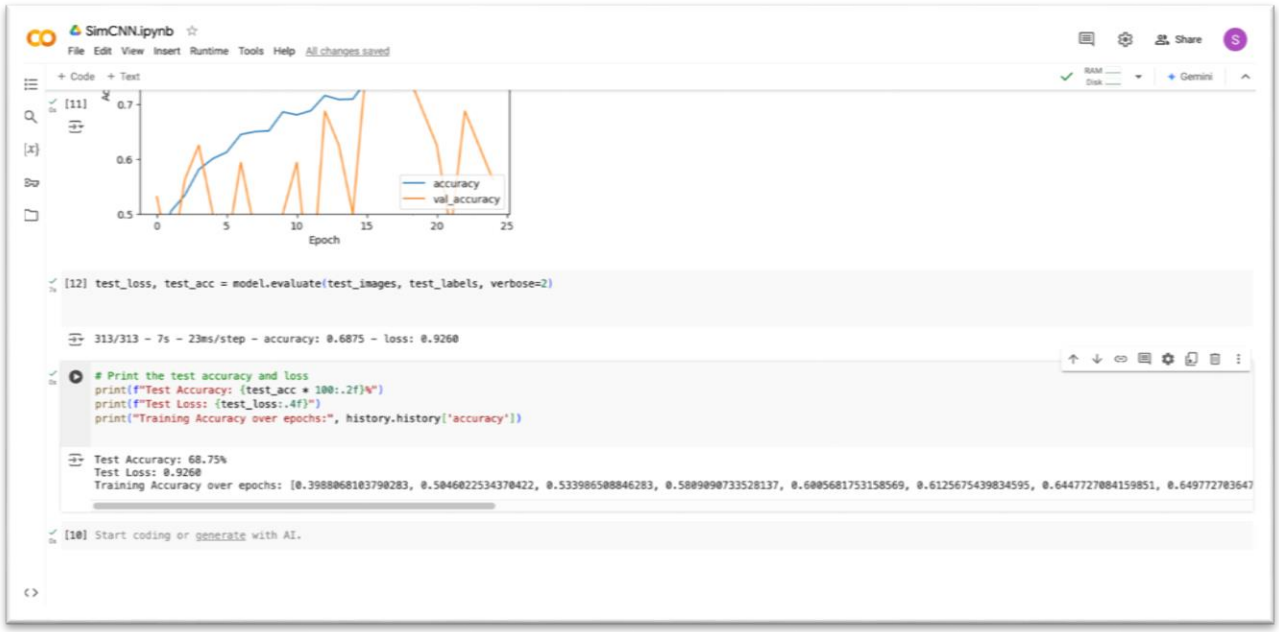
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
batch_normalization (BatchNormalization)	(None, 30, 30, 32)	128
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 13, 13, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36,928
batch_normalization_2 (BatchNormalization)	(None, 4, 4, 64)	256
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65,600
dense_1 (Dense)	(None, 10)	650

Total params: 123,210 (481.29 KB)
Trainable params: 122,890 (480.04 KB)
Non-trainable params: 320 (1.25 KB)



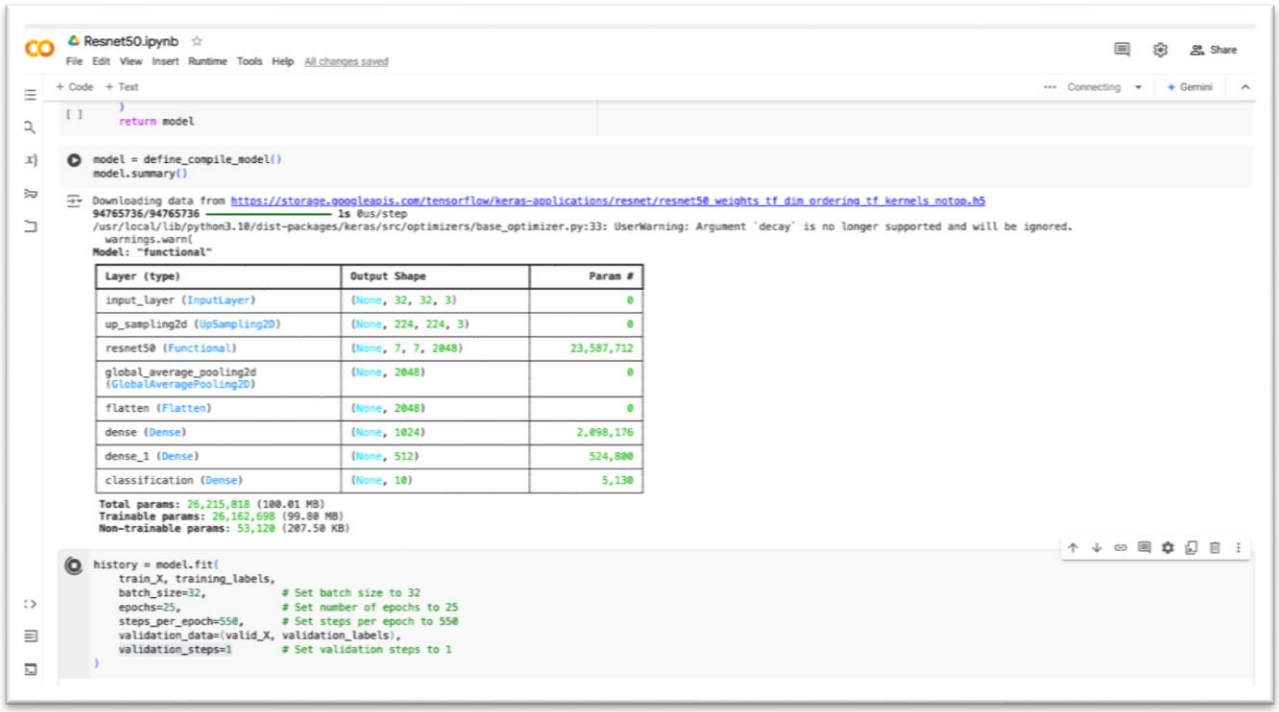
```
/usr/local/lib/python3.10/dist-packages/keras/src/optimizers/base_optimizer.py:33: UserWarning: Argument `decay` is no longer supported and will be ignored.
  warnings.warn(

# Train the model
history = model.fit(
    train_images, train_labels,
    batch_size=32,
    epochs=25,
    steps_per_epoch = ndarray with shape (10000, 32, 32, 3)
    validation_data=(test_images, test_labels),
    validation_steps=1
)

Epoch 1/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 43s 75ms/step - accuracy: 0.3268 - loss: 1.9337 - val_accuracy: 0.5312 - val_loss: 1.2485
Epoch 2/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 33s 59ms/step - accuracy: 0.4889 - loss: 1.4415 - val_accuracy: 0.4062 - val_loss: 1.7755
Epoch 3/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 28s 50ms/step - accuracy: 0.5253 - loss: 1.3299 - val_accuracy: 0.5625 - val_loss: 1.2026
Epoch 4/25
/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch *
  self.gen.throw(typ, value, traceback)
550/550 ━━━━━━━━━━━━━━━━━━━━ 75s 63ms/step - accuracy: 0.5797 - loss: 1.1883 - val_accuracy: 0.6250 - val_loss: 1.2571
Epoch 5/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 33s 60ms/step - accuracy: 0.5938 - loss: 1.1517 - val_accuracy: 0.5000 - val_loss: 1.9031
Epoch 6/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 28s 51ms/step - accuracy: 0.6102 - loss: 1.1154 - val_accuracy: 0.4375 - val_loss: 1.6175
Epoch 7/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 32s 58ms/step - accuracy: 0.6410 - loss: 1.0155 - val_accuracy: 0.5938 - val_loss: 1.2407
Epoch 8/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 32s 59ms/step - accuracy: 0.6565 - loss: 0.9945 - val_accuracy: 0.4688 - val_loss: 1.4054
Epoch 9/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 26s 48ms/step - accuracy: 0.6525 - loss: 0.9817 - val_accuracy: 0.3125 - val_loss: 1.6202
Epoch 10/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 82s 55ms/step - accuracy: 0.6813 - loss: 0.9003 - val_accuracy: 0.5000 - val_loss: 1.2104
Epoch 11/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 32s 57ms/step - accuracy: 0.6820 - loss: 0.9045 - val_accuracy: 0.5938 - val_loss: 0.9200
Epoch 12/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 26s 47ms/step - accuracy: 0.6819 - loss: 0.8948 - val_accuracy: 0.3438 - val_loss: 1.8166
Epoch 13/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 88s 62ms/step - accuracy: 0.7150 - loss: 0.8043 - val_accuracy: 0.6875 - val_loss: 0.9839
Epoch 14/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 33s 59ms/step - accuracy: 0.7034 - loss: 0.8393 - val_accuracy: 0.6250 - val_loss: 1.1074
```





RESNET50 : filename :RESNET50_01.py

Using the training and testing data from CIFAR-10, perform classification with ResNet50 and find the classification accuracy with parameters in table 1



Resnet50.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

Connecting

Gemini

```
    )
    return model
```

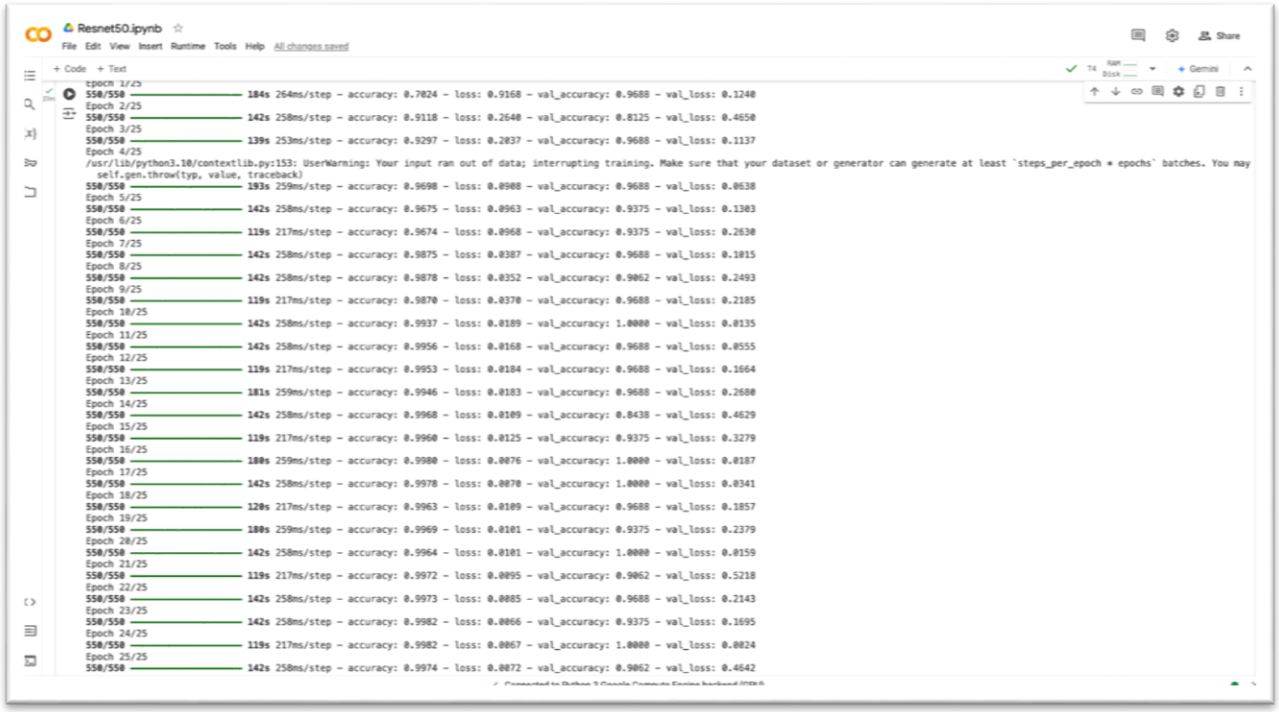
```
model = define_compile_model()
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/optimizers/base_optimizer.py:33: UserWarning: Argument `decay` is no longer supported and will be ignored.
  warnings.warn(
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 32, 32, 3)	0
up_sampling2d (UpSampling2D)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dense_1 (Dense)	(None, 512)	524,800
classification (Dense)	(None, 10)	5,130

Total params: 26,215,818 (100.01 MB)
Trainable params: 26,162,698 (99.80 MB)
Non-trainable params: 53,120 (207.50 KB)

```
history = model.fit(
    train_X, training_labels,
    batch_size=32,              # Set batch size to 32
    epochs=25,                  # Set number of epochs to 25
    steps_per_epoch=550,        # Set steps per epoch to 550
    validation_data=(valid_X, validation_labels),
    validation_steps=1          # Set validation steps to 1
)
```



Resnet50.ipynb

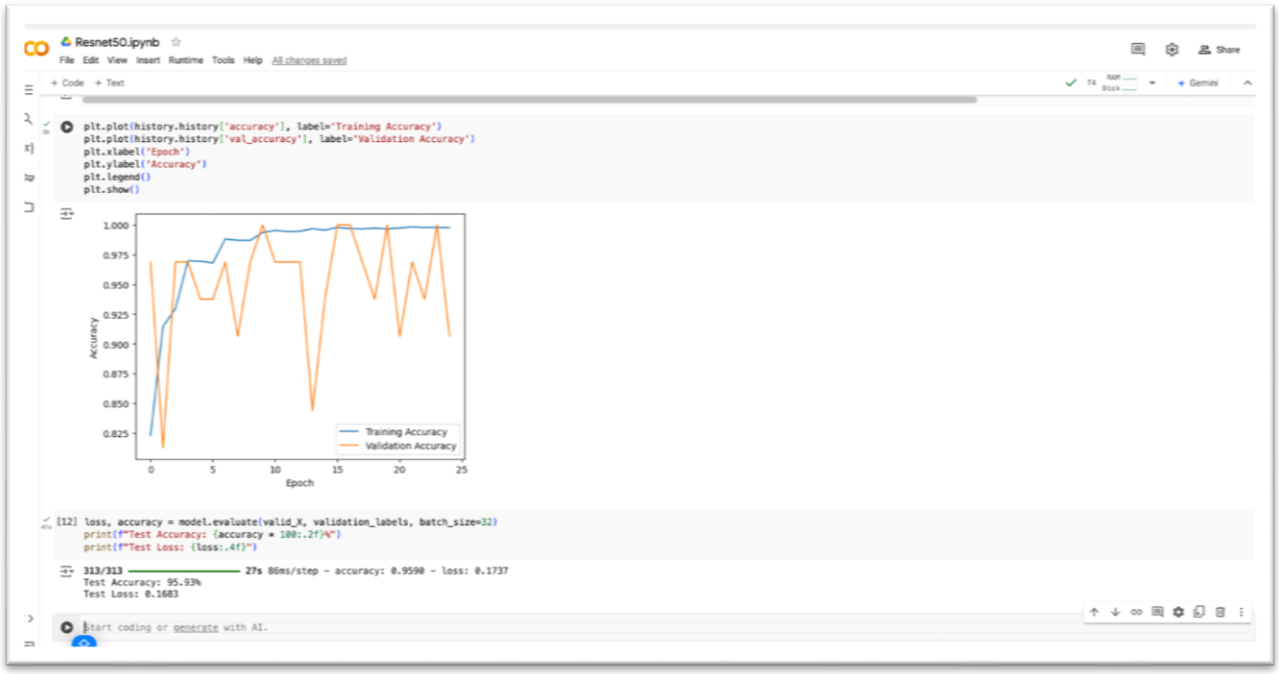
File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

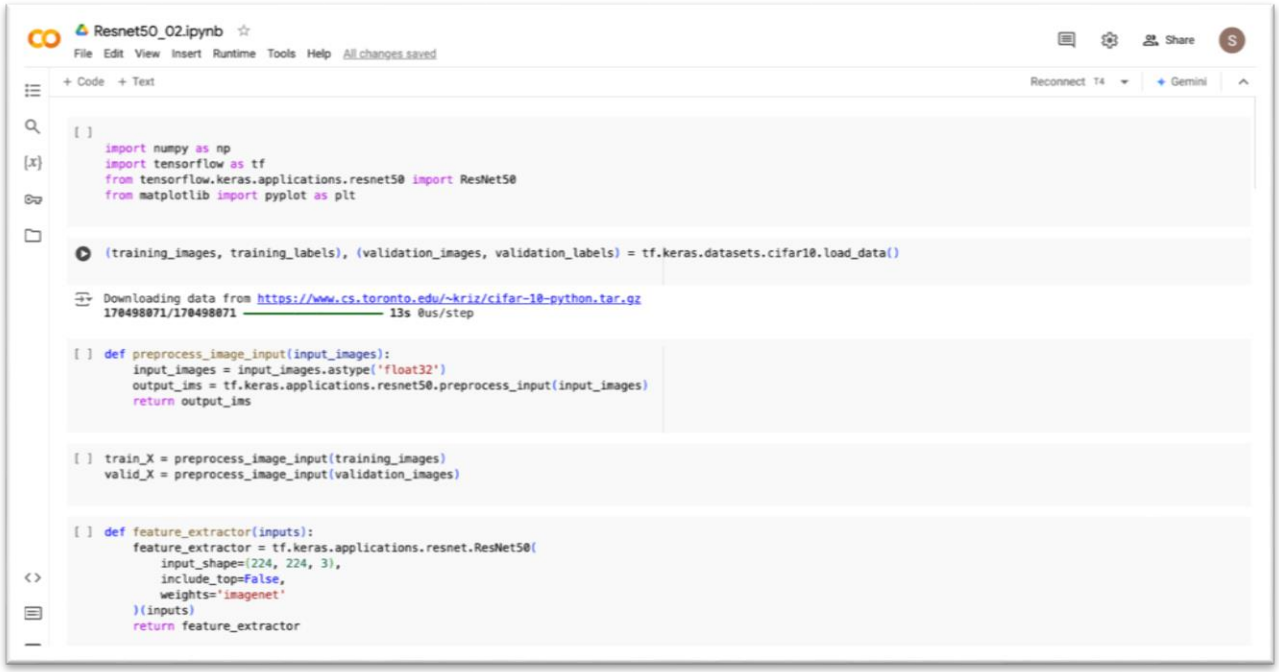
Gemini

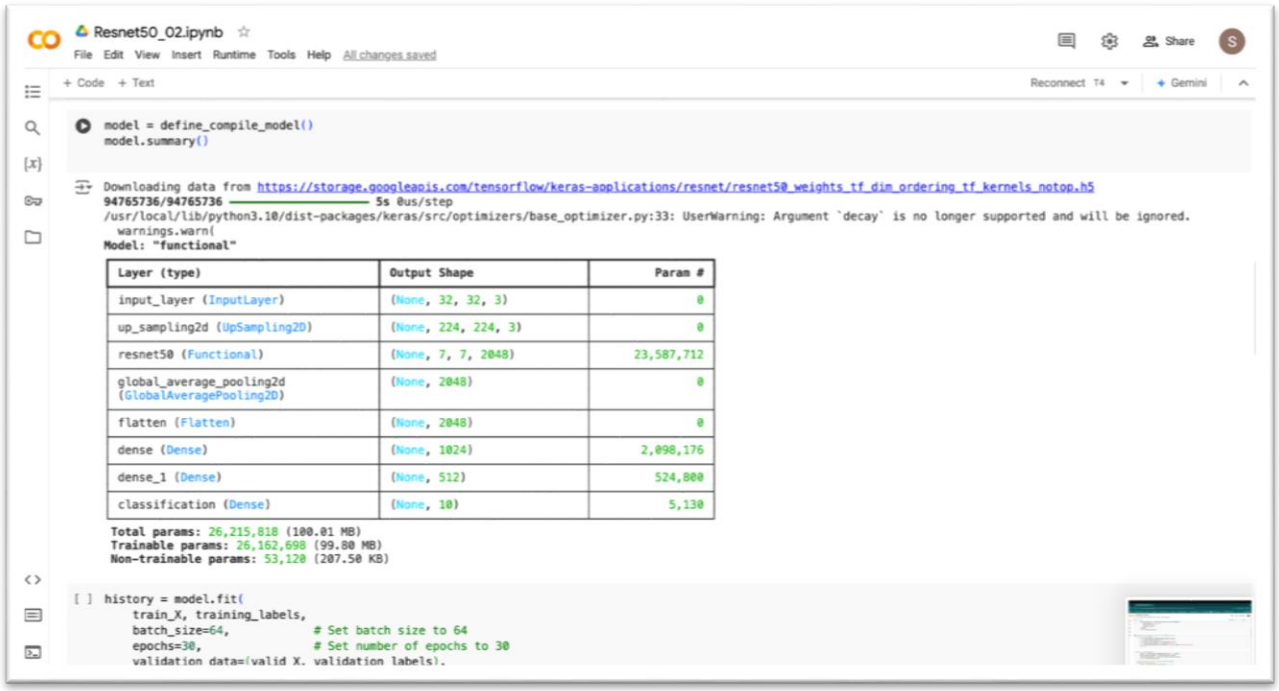
```
Epoch 1/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 184s 264ms/step - accuracy: 0.7024 - loss: 0.9168 - val_accuracy: 0.9688 - val_loss: 0.1240
Epoch 2/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9118 - loss: 0.2640 - val_accuracy: 0.8125 - val_loss: 0.4650
Epoch 3/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 139s 253ms/step - accuracy: 0.9297 - loss: 0.2037 - val_accuracy: 0.9688 - val_loss: 0.1137
Epoch 4/25
/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches. You may
  self.gen.throw(typ, value, traceback)
550/550 ━━━━━━━━━━━━━━━━━━━━ 193s 259ms/step - accuracy: 0.9698 - loss: 0.0908 - val_accuracy: 0.9688 - val_loss: 0.0638
Epoch 5/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9675 - loss: 0.0963 - val_accuracy: 0.9375 - val_loss: 0.1303
Epoch 6/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 119s 217ms/step - accuracy: 0.9674 - loss: 0.0968 - val_accuracy: 0.9375 - val_loss: 0.2630
Epoch 7/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9875 - loss: 0.0387 - val_accuracy: 0.9688 - val_loss: 0.1015
Epoch 8/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9878 - loss: 0.0352 - val_accuracy: 0.9062 - val_loss: 0.2493
Epoch 9/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 119s 217ms/step - accuracy: 0.9870 - loss: 0.0370 - val_accuracy: 0.9688 - val_loss: 0.2185
Epoch 10/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9937 - loss: 0.0189 - val_accuracy: 1.0000 - val_loss: 0.0135
Epoch 11/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9956 - loss: 0.0168 - val_accuracy: 0.9688 - val_loss: 0.0555
Epoch 12/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 119s 217ms/step - accuracy: 0.9953 - loss: 0.0184 - val_accuracy: 0.9688 - val_loss: 0.1664
Epoch 13/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 181s 259ms/step - accuracy: 0.9946 - loss: 0.0183 - val_accuracy: 0.9688 - val_loss: 0.2680
Epoch 14/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9968 - loss: 0.0109 - val_accuracy: 0.8438 - val_loss: 0.4629
Epoch 15/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 119s 217ms/step - accuracy: 0.9960 - loss: 0.0125 - val_accuracy: 0.9375 - val_loss: 0.3279
Epoch 16/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 180s 259ms/step - accuracy: 0.9980 - loss: 0.0076 - val_accuracy: 1.0000 - val_loss: 0.0187
Epoch 17/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9978 - loss: 0.0070 - val_accuracy: 1.0000 - val_loss: 0.0341
Epoch 18/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 120s 217ms/step - accuracy: 0.9963 - loss: 0.0109 - val_accuracy: 0.9688 - val_loss: 0.1857
Epoch 19/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 180s 259ms/step - accuracy: 0.9969 - loss: 0.0101 - val_accuracy: 0.9375 - val_loss: 0.2379
Epoch 20/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9964 - loss: 0.0101 - val_accuracy: 1.0000 - val_loss: 0.0159
Epoch 21/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 119s 217ms/step - accuracy: 0.9972 - loss: 0.0095 - val_accuracy: 0.9062 - val_loss: 0.5218
Epoch 22/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9973 - loss: 0.0085 - val_accuracy: 0.9688 - val_loss: 0.2143
Epoch 23/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9982 - loss: 0.0066 - val_accuracy: 0.9375 - val_loss: 0.1695
Epoch 24/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 119s 217ms/step - accuracy: 0.9982 - loss: 0.0067 - val_accuracy: 1.0000 - val_loss: 0.0024
Epoch 25/25
550/550 ━━━━━━━━━━━━━━━━━━━━ 142s 258ms/step - accuracy: 0.9974 - loss: 0.0072 - val_accuracy: 0.9062 - val_loss: 0.4642
```



RESNET50 With Own Parameter Values & Justification:

filename: RESNET50_02.py





Resnet50_02.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

Reconnect T4 Gemini

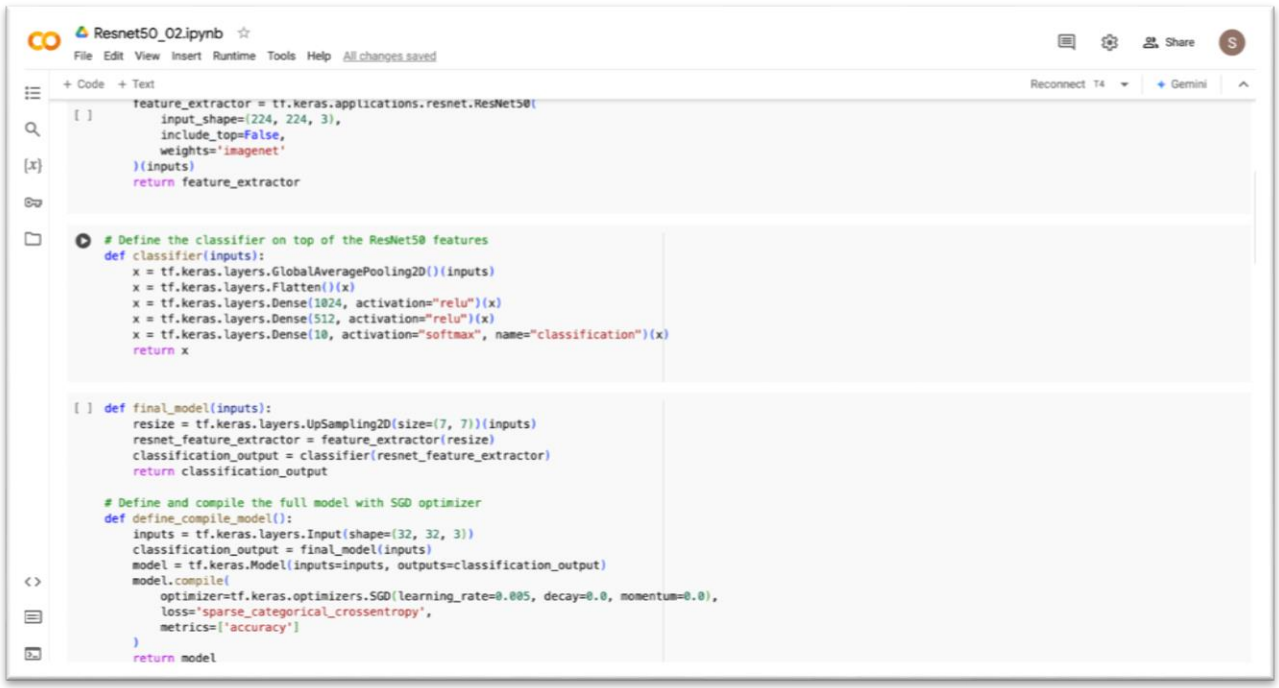
```
model = define_compile_model()
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/optimizers/base_optimizer.py:33: UserWarning: Argument `decay` is no longer supported and will be ignored.
  warnings.warn(
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 32, 32, 3)	0
up_sampling2d (UpSampling2D)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dense_1 (Dense)	(None, 512)	524,800
classification (Dense)	(None, 10)	5,130

```
Total params: 26,215,818 (100.01 MB)
Trainable params: 26,162,698 (99.80 MB)
Non-trainable params: 53,120 (207.50 KB)
```

```
[ ] history = model.fit(
        train_X, training_labels,
        batch_size=64,            # Set batch size to 64
        epochs=30,                # Set number of epochs to 30
        validation_data=(valid_X, validation_labels),
```



Resnet50_02.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

Reconnect T4 Gemini

```
[ ]     feature_extractor = tf.keras.applications.resnet.ResNet50(
            input_shape=(224, 224, 3),
            include_top=False,
            weights='imagenet'
        )(inputs)
        return feature_extractor
```

```
# Define the classifier on top of the ResNet50 features
def classifier(inputs):
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dense(1024, activation="relu")(x)
    x = tf.keras.layers.Dense(512, activation="relu")(x)
    x = tf.keras.layers.Dense(10, activation="softmax", name="classification")(x)
    return x
```

```
[ ] def final_model(inputs):
        resize = tf.keras.layers.UpSampling2D(size=(7, 7))(inputs)
        resnet_feature_extractor = feature_extractor(resize)
        classification_output = classifier(resnet_feature_extractor)
        return classification_output

    # Define and compile the full model with SGD optimizer
    def define_compile_model():
        inputs = tf.keras.layers.Input(shape=(32, 32, 3))
        classification_output = final_model(inputs)
        model = tf.keras.Model(inputs=inputs, outputs=classification_output)
        model.compile(
            optimizer=tf.keras.optimizers.SGD(learning_rate=0.005, decay=0.0, momentum=0.0),
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy']
        )
        return model
```

Resnet50_02.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

Reconnect T4 Gemini

```
history = model.fit(
    train_X, training_labels,
    batch_size=64,            # Set batch size to 64
    epochs=30,                # Set number of epochs to 30
    validation_data=(valid_X, validation_labels),
    validation_steps=1        # Set validation steps to 1
)
```

```
Epoch 1/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 482s 559ms/step - accuracy: 0.7003 - loss: 0.9587 - val_accuracy: 0.8750 - val_loss: 0.2346
Epoch 2/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 445s 528ms/step - accuracy: 0.9474 - loss: 0.1648 - val_accuracy: 0.9688 - val_loss: 0.1119
Epoch 3/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 528ms/step - accuracy: 0.9752 - loss: 0.0792 - val_accuracy: 0.9531 - val_loss: 0.1897
Epoch 4/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9901 - loss: 0.0385 - val_accuracy: 0.9531 - val_loss: 0.2114
Epoch 5/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9961 - loss: 0.0203 - val_accuracy: 0.9531 - val_loss: 0.2240
Epoch 6/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9983 - loss: 0.0121 - val_accuracy: 0.9531 - val_loss: 0.2113
Epoch 7/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 412s 527ms/step - accuracy: 0.9989 - loss: 0.0084 - val_accuracy: 0.9219 - val_loss: 0.2516
Epoch 8/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9991 - loss: 0.0068 - val_accuracy: 0.9688 - val_loss: 0.1805
Epoch 9/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9995 - loss: 0.0048 - val_accuracy: 0.9531 - val_loss: 0.2515
Epoch 10/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9993 - loss: 0.0051 - val_accuracy: 0.9688 - val_loss: 0.1760
Epoch 11/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9996 - loss: 0.0033 - val_accuracy: 0.9531 - val_loss: 0.1940
Epoch 12/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 412s 527ms/step - accuracy: 0.9999 - loss: 0.0024 - val_accuracy: 0.9688 - val_loss: 0.0841
Epoch 13/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9996 - loss: 0.0027 - val_accuracy: 0.9531 - val_loss: 0.1175
Epoch 14/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 411s 526ms/step - accuracy: 0.9993 - loss: 0.0031 - val_accuracy: 0.9062 - val_loss: 0.4498
Epoch 15/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 443s 527ms/step - accuracy: 0.9998 - loss: 0.0022 - val_accuracy: 0.9375 - val_loss: 0.2784
Epoch 16/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9997 - loss: 0.0024 - val_accuracy: 0.9844 - val_loss: 0.0648
Epoch 17/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9999 - loss: 0.0016 - val_accuracy: 0.9688 - val_loss: 0.1022
Epoch 18/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 528ms/step - accuracy: 1.0000 - loss: 0.0014 - val_accuracy: 0.9531 - val_loss: 0.1989
Epoch 19/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 441s 527ms/step - accuracy: 0.9998 - loss: 0.0012 - val_accuracy: 0.9531 - val_loss: 0.2872
Epoch 20/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 528ms/step - accuracy: 1.0000 - loss: 9.8054e-04 - val_accuracy: 0.9375 - val_loss: 0.3719
```

Resnet50_02.ipynb

File Edit View Insert Runtime Tools Help All changes saved

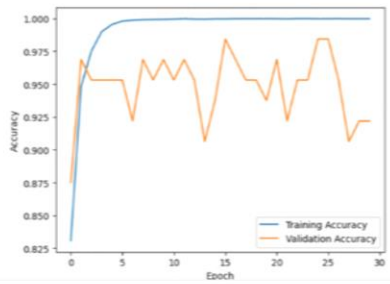
Share

+ Code + Text

Reconnect T4 Gemini

```
Epoch 22/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9998 - loss: 0.0015 - val_accuracy: 0.9219 - val_loss: 0.2635
Epoch 23/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 1.0000 - loss: 0.0012 - val_accuracy: 0.9531 - val_loss: 0.0767
Epoch 24/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 1.0000 - loss: 7.9174e-04 - val_accuracy: 0.9531 - val_loss: 0.2741
Epoch 25/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9999 - loss: 9.4677e-04 - val_accuracy: 0.9844 - val_loss: 0.0641
Epoch 26/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9999 - loss: 8.5237e-04 - val_accuracy: 0.9844 - val_loss: 0.0522
Epoch 27/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 412s 526ms/step - accuracy: 0.9999 - loss: 8.3829e-04 - val_accuracy: 0.9531 - val_loss: 0.1530
Epoch 28/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 443s 527ms/step - accuracy: 0.9997 - loss: 9.0868e-04 - val_accuracy: 0.9062 - val_loss: 0.5167
Epoch 29/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 1.0000 - loss: 7.5948e-04 - val_accuracy: 0.9219 - val_loss: 0.1528
Epoch 30/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 442s 527ms/step - accuracy: 0.9999 - loss: 9.3109e-04 - val_accuracy: 0.9219 - val_loss: 0.2067
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
loss, accuracy = model.evaluate(valid_X, validation_labels, batch_size=64)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
print(f"Test Loss: {loss:.4f}")
```

```
157/157 ━━━━━━━━━━━━━━━━━━━━ 26s 165ms/step - accuracy: 0.9467 - loss: 0.2654
Test Accuracy: 94.67%
Test Loss: 0.2565
```

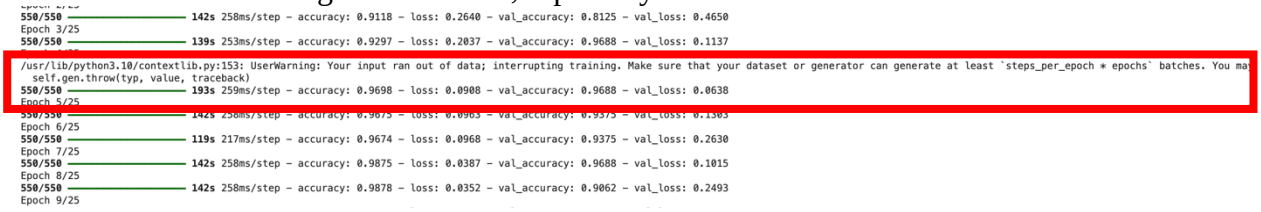
JUSTIFICATION FOR OWN PARAMATER VALUES

Batch Size = 64: Increasing to 64 improves computation efficiency by processing more samples per batch, which stabilizes gradient updates and speeds up training without sacrificing accuracy on the small CIFAR-10 dataset.

Epochs = 30: Slightly increasing epochs gives the model more time to learn features, improving accuracy without risking overfitting, especially with a pretrained ResNet50.

Learning Rate (0.005): Lowering it from 0.01 allows for more stable and precise updates.

Steps per Epoch : Automatically calculating `steps_per_epoch` based on the dataset size and batch size ensures that each epoch processes the entire training set. This prevents `OUT_OF_RANGE` errors and makes training more reliable, especially with small datasets like CIFAR-10.



Summary of Modifications and Benefits

- **Batch Size**: Increased to reduce training time per epoch.
- **Epochs**: Increased slightly to improve accuracy with more learning iterations.
- **Removing** Steps per Epoch
- **SGD Parameters**:
    - **Learning Rate**: Lowered for more stable training.

Explanation of 782 Steps per Epoch

1. **Dataset Size**: CIFAR-10's training set contains **50,000 images**.
2. **Batch Size**: In your code, you've set the batch size to **64**.
3. **Steps per Epoch Calculation**:
    - The number of steps per epoch is calculated as the total number of training samples divided by the batch size.
    - Formula: steps_per_epoch=total samples/batch size=50,000/64≈781.25
    - Since steps per epoch must be an integer, Keras rounds up to **782** steps to ensure all samples are covered in each epoch.

(a) Compare the classification accuracies of SimCNN and ResNet50 using the parameters in Table 1 of the paper "Transfer_Learning_Based_On_ResNet50". This paper is posted on Canvas under Lecture 6 and Project Assignment 1 folders. Choose parameters from Table 1 as appropriate for each model. Note that parameters and hyperparameters are used interchangeably. [40 points]

1. **SimCNN**:
    - **Performance**:
        - **Test Accuracy**: 68.75%
        - **Test Loss**: 0.9260
2. **ResNet50**:
    - **Performance**:
        - **Test Accuracy**: 95.93%
        - **Test Loss**: 0.1683

Key Differences

- **Accuracy**: ResNet50 was **27.18% more accurate** than SimCNN, with almost 96% accuracy on the test set compared to SimCNN's 68.75%.
- **Loss**: ResNet50 had a much lower loss, showing it made fewer mistakes overall.
- **Training Speed**: ResNet50 learned faster and reached high accuracy quicker, thanks to its pre-trained layers and advanced design. SimCNN took more training steps to get to moderate accuracy.
-

(b) Compare the classification accuracies of ResNet50 implemented in (a), with ResNet50 implemented with your own set of parameter values. Justify your choices of parameters. [40 points]

1. **ResNet50-01 (Default Parameters)**:
    - **Batch Size**: 32
    - **Epochs**: 25
    - **Steps per Epoch**: 550
    - **Validation Steps**: 1
    - **Test Accuracy**: 95.93%
    - **Test Loss**: 0.1683
2. **ResNet50-02 (Custom Parameters)**:
    - **Batch Size**: 64
    - **Epochs**: 30
    - **Validation Steps**: 1
    - **Test Accuracy**: 94.67%
    - **Test Loss**: 0.2565

Test Accuracy:

- **ResNet50-01** achieved a slightly higher test accuracy of **95.93%**, compared to **94.67%** with ResNet50-02.

Test Loss:

- **ResNet50-01** also had a lower test loss (0.1683) compared to ResNet50-02 (0.2565), indicating it made fewer errors on the test set.
- The lower loss in ResNet50-01 suggests it may have had a slightly better fit to the data compared to the custom parameter set.

Training Efficiency:

• ResNet50-02 reduced the number of updates per epoch and increased training speed by using a bigger batch size of 64. It also had a longer training period—30 epochs—which might have allowed it to pick up more patterns.

• ResNet50-02's accuracy was not higher than ResNet50-01's, even with the additional epochs. This could mean that the model was not greatly improved by adding more epochs and a larger batch size, or that the default setup (ResNet50-01) was more appropriate for this purpose.

CONCLUSION:

The below table summarizes the findings in each case :

Model	Batch Size	Epochs	Test Accuracy	Test Loss
SimCNN	32	25	68.75%	0.9260
ResNet50-01	32	25	95.93%	0.1683
ResNet50-02	64	30	94.67%	0.2565

Best Model: ResNet50-01 with default parameters provided the best classification accuracy and lowest test loss, indicating it was well-suited to the CIFAR-10 dataset with minimal adjustments.

Recommendation: For tasks requiring high accuracy on CIFAR-10-like datasets, a pre-trained model like ResNet50 with moderate batch sizes and epochs is recommended.