Spring 2025: Neural Networks & Deep Learning – ICP -5

Assignment -5

Name: Vanitha Chintalapudi

Student ID: 700756782

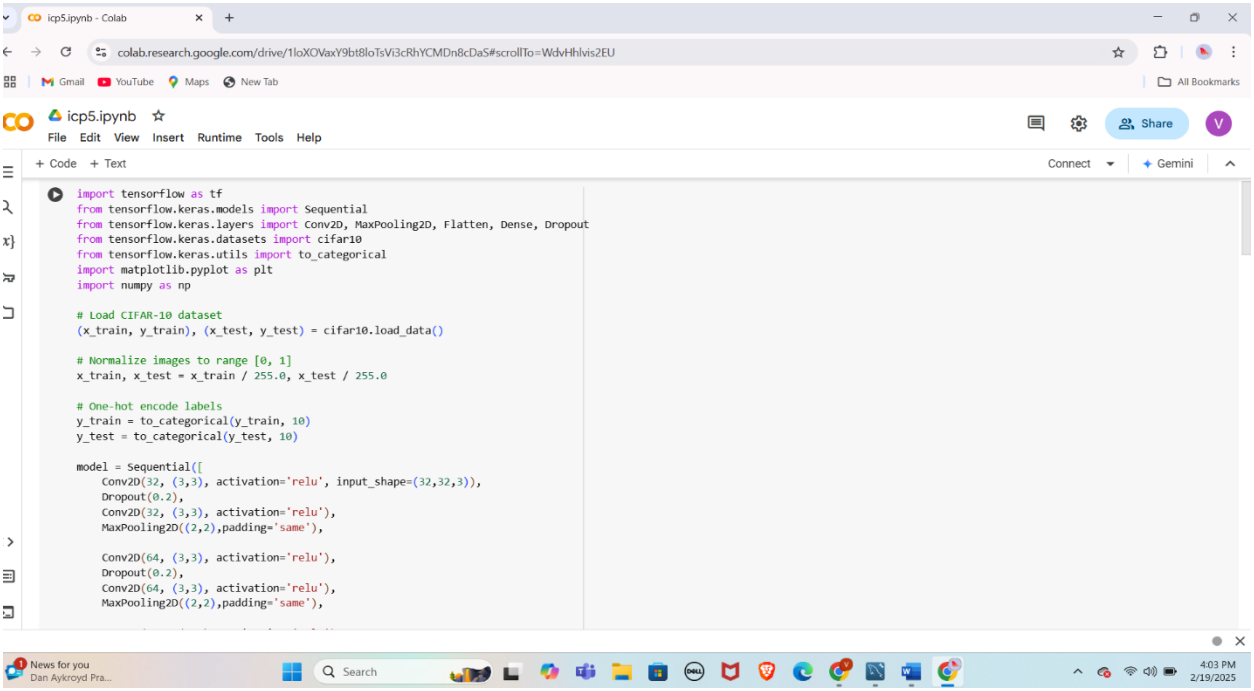
Github Link: https://github.com/VanithaChintalapudi10/Neural-network-deep-learning

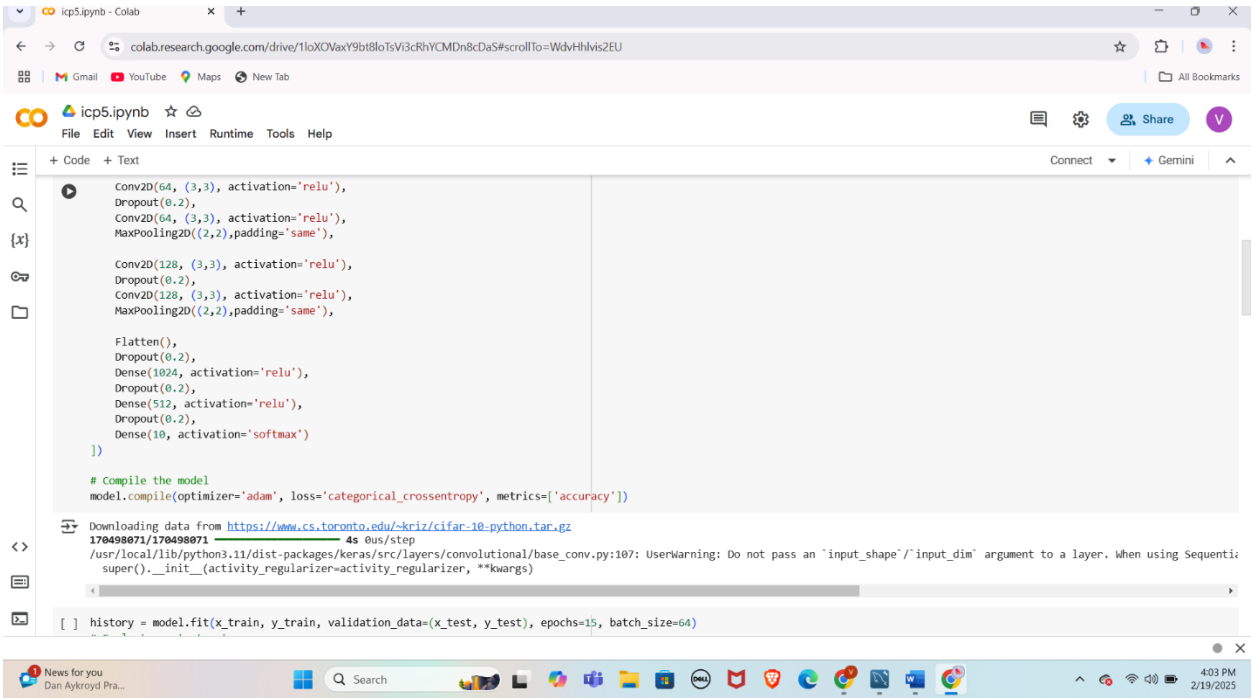
Video Link: https://drive.google.com/file/d/1Ow8I1IBJn2_Lyb9KH_ZBlrE3j6wQubW/view?usp=drive_link

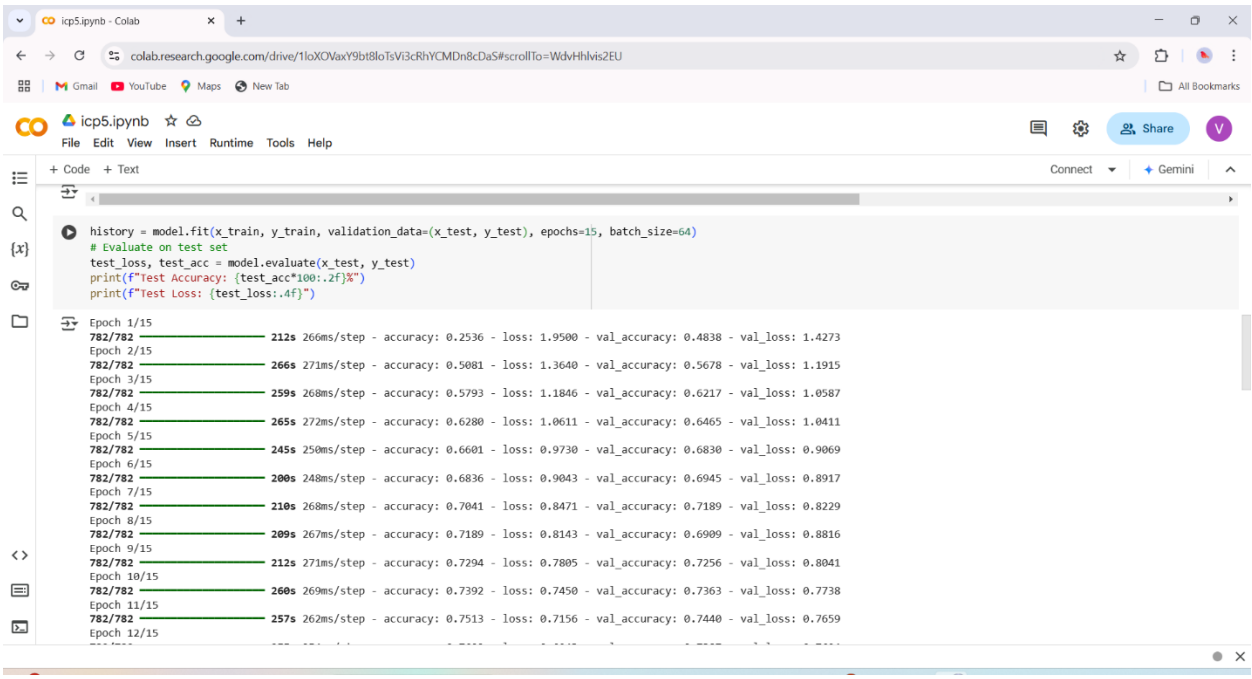
1. Follow the instruction below and then report how the performance changed.(apply all at once)
   • Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
   • Dropout layer at 20%.
   • Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
   • Max Pool layer with size 2×2.
   • Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
   • Dropout layer at 20%.
   • Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
   • Max Pool layer with size 2×2.
   • Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
   • Dropout layer at 20%.
   • Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function.
   • Max Pool layer with size 2×2.
   • Flatten layer.
   • Dropout layer at 20%.
   • Fully connected layer with 1024 units and a rectifier activation function.

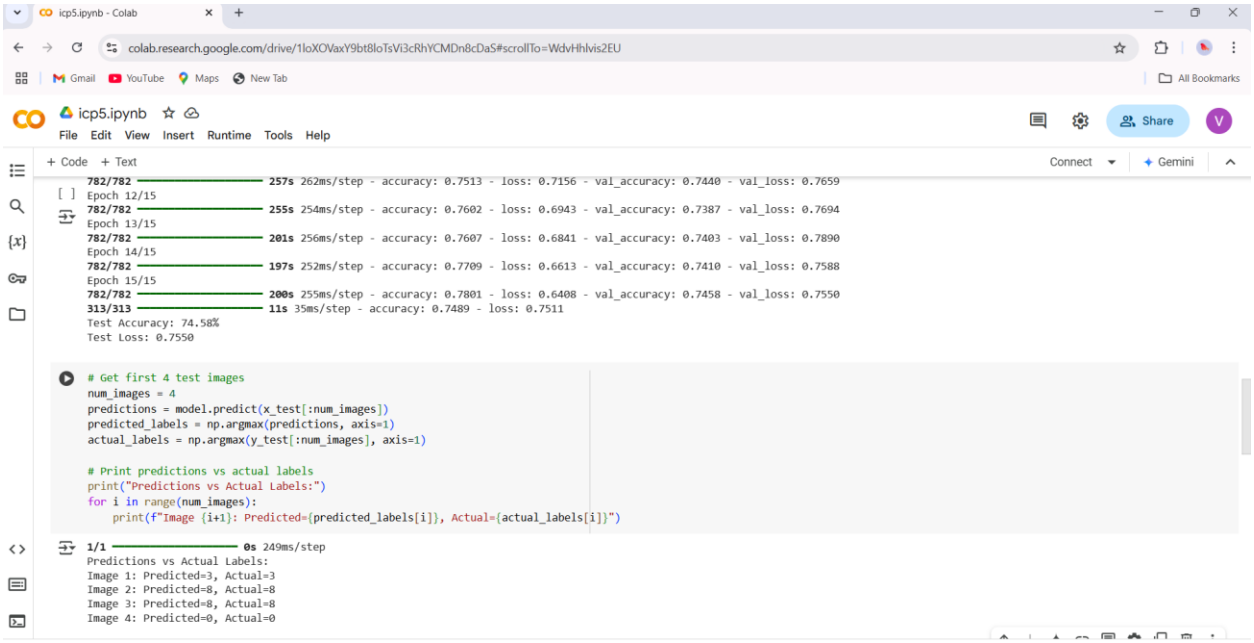
- Dropout layer at 20%.
- Fully connected layer with 512 units and a rectifier activation function.
- Dropout layer at 20%.
- Fully connected output layer with 10 units and a Softmax activation function

Output:

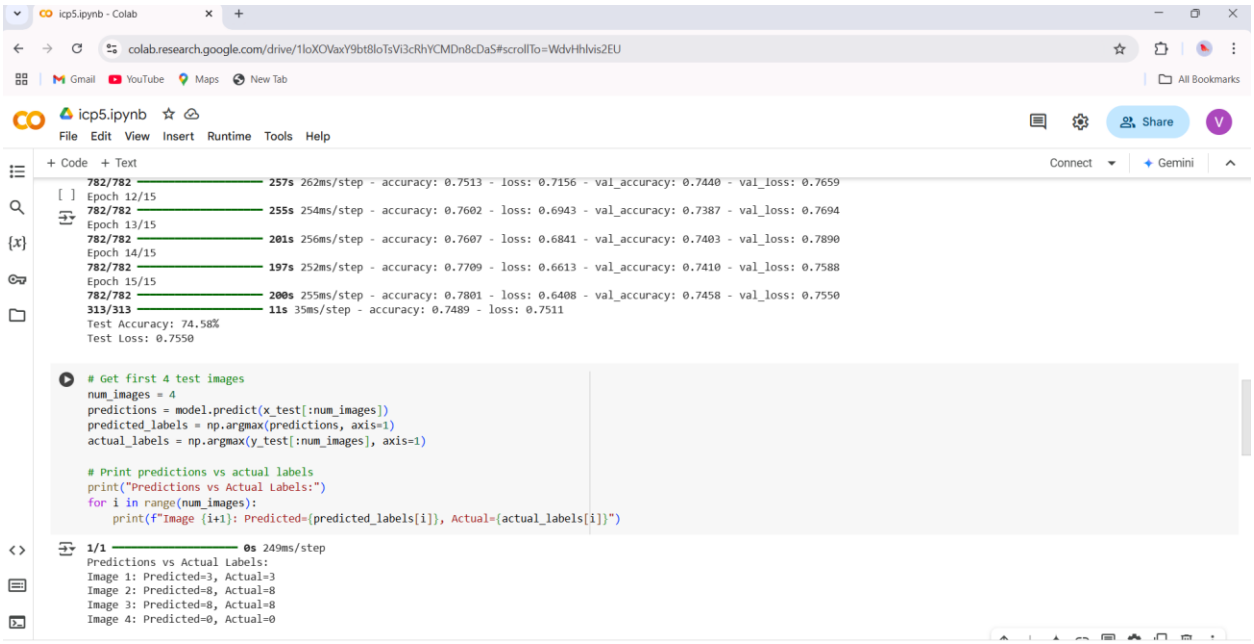



icp5.ipynb - Colab
colab.research.google.com/drive/1loXOVaxY9bt8loTsVi3cRhYCMDn8cDaS#scrollTo=WdvHhlvis2EU
icp5.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Connect
Gemini
Conv2D(64, (3,3), activation='relu'),
Dropout(0.2),
Conv2D(64, (3,3), activation='relu'),
MaxPooling2D((2,2),padding='same'),
Conv2D(128, (3,3), activation='relu'),
Dropout(0.2),
Conv2D(128, (3,3), activation='relu'),
MaxPooling2D((2,2),padding='same'),
Flatten(),
Dropout(0.2),
Dense(1024, activation='relu'),
Dropout(0.2),
Dense(512, activation='relu'),
Dropout(0.2),
Dense(10, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 4s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequentia
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
[ ] history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=15, batch_size=64)
News for you
Dan Aykroyd Pra...
Search
4:03 PM
2/19/2025


icp5.ipynb - Colab
colab.research.google.com/drive/1loXOVaxY9bt8loTsVi3cRhYCMDn8cDaS#scrollTo=WdvHhlvis2EU
icp5.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Connect
Gemini
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=15, batch_size=64)
# Evaluate on test set
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc*100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")
Epoch 1/15
782/782 212s 266ms/step - accuracy: 0.2536 - loss: 1.9500 - val_accuracy: 0.4838 - val_loss: 1.4273
Epoch 2/15
782/782 266s 271ms/step - accuracy: 0.5081 - loss: 1.3640 - val_accuracy: 0.5678 - val_loss: 1.1915
Epoch 3/15
782/782 259s 268ms/step - accuracy: 0.5793 - loss: 1.1846 - val_accuracy: 0.6217 - val_loss: 1.0587
Epoch 4/15
782/782 265s 272ms/step - accuracy: 0.6280 - loss: 1.0611 - val_accuracy: 0.6465 - val_loss: 1.0411
Epoch 5/15
782/782 245s 250ms/step - accuracy: 0.6601 - loss: 0.9730 - val_accuracy: 0.6830 - val_loss: 0.9069
Epoch 6/15
782/782 200s 248ms/step - accuracy: 0.6836 - loss: 0.9043 - val_accuracy: 0.6945 - val_loss: 0.8917
Epoch 7/15
782/782 210s 268ms/step - accuracy: 0.7041 - loss: 0.8471 - val_accuracy: 0.7189 - val_loss: 0.8229
Epoch 8/15
782/782 209s 267ms/step - accuracy: 0.7189 - loss: 0.8143 - val_accuracy: 0.6909 - val_loss: 0.8816
Epoch 9/15
782/782 212s 271ms/step - accuracy: 0.7294 - loss: 0.7805 - val_accuracy: 0.7256 - val_loss: 0.8041
Epoch 10/15
782/782 260s 269ms/step - accuracy: 0.7392 - loss: 0.7450 - val_accuracy: 0.7363 - val_loss: 0.7738
Epoch 11/15
782/782 257s 262ms/step - accuracy: 0.7513 - loss: 0.7156 - val_accuracy: 0.7440 - val_loss: 0.7659
Epoch 12/15



2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.



3. Visualize Loss and Accuracy using the history object

Output: