CSE425

Assignment 2

Muhammad Nayeem Mubasshirul Haque

ID: 19101115

Section:03

Custom Convolutional Neural Network Implementation and Comparative Analysis

Introduction

In this report, the implementation and analysis of a custom Convolutional Neural Network (CNN) architecture for image classification tasks using the CIFAR-10 dataset are done. The primary objective is to design, train, and rigorously evaluate this custom CNN model using various configurations and techniques, focusing on its performance characteristics and limitations.

Dataset

The CIFAR-10 dataset, consisting of 60,000 32x32 color images in 10 classes, was used for training and testing the custom CNN models. The dataset was preprocessed to normalize pixel values and split into training and validation sets. The dataset further distinguishes itself by categorizing these images into precisely 10 classes, spanning a diverse spectrum of objects and concepts.

The CIFAR-10 dataset contains a wide variety of objects and concepts, classified into 10 distinct categories:

1. Airplanes
2. Automobiles
3. Birds
4. Cats
5. Deer
6. Dogs
7. Frogs
8. Horses
9. Ships
10. Trucks

Model Architecture

Constructed using the TensorFlow and Keras libraries, the custom CNN architecture comprises a stratified sequence of convolutional and pooling layers. This foundational structure is subsequently followed by fully connected layers and a final softmax output layer, facilitating effective classification.

Implementing Traditional CNNs

The initial phase involves the development of a traditional CNN architecture featuring alternating convolutional and pooling layers. The model achieved a commendable validation accuracy of approximately 70%. The model configuration and results are shown below:

Filters: 32

Kernel Size: (3, 3)

Pooling: MaxPooling (2, 2)

Activation Function: ReLU

Training Accuracy: ~79%

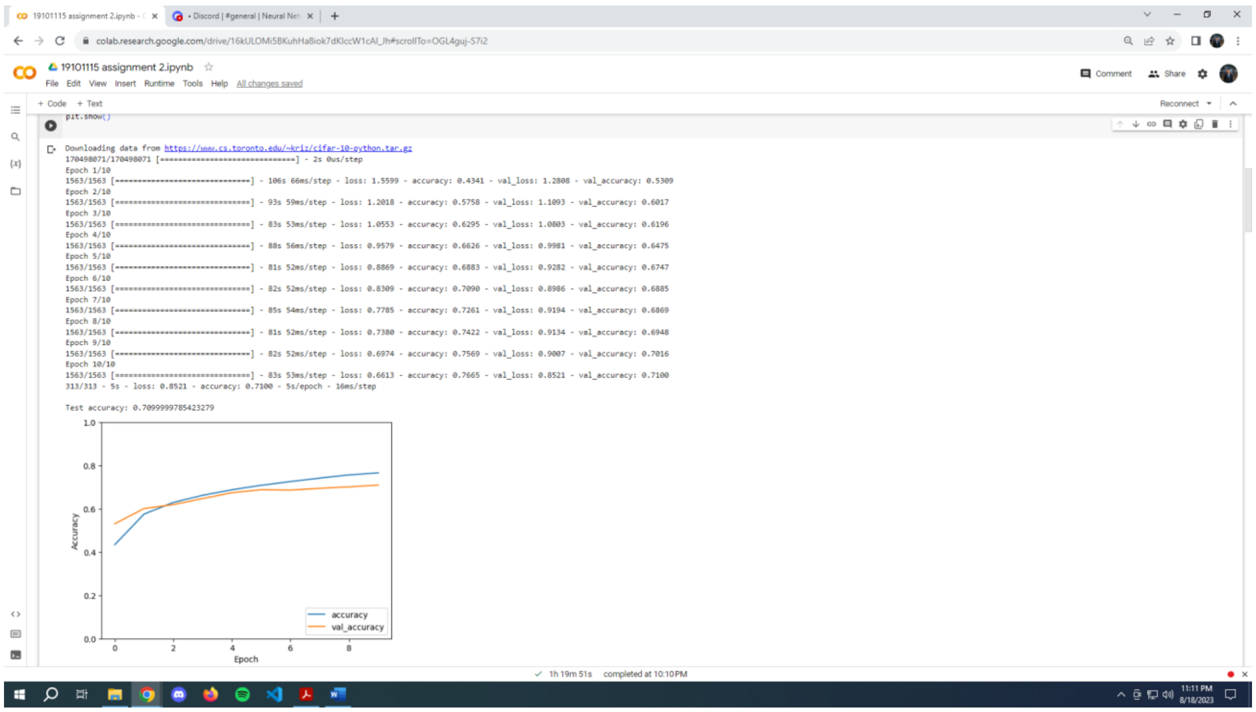
Validation Accuracy: ~70%

CODE:

```
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
import matplotlib.pyplot as plt

# Load and preprocess CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Define the CNN architecture
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
```

```python
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train, epochs=10,
                    validation_data=(x_test, y_test))

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print("\nTest accuracy:", test_acc)

# Plot training and validation accuracy over epochs
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

Output:


19101115 assignment 2.ipynb
Discord | #general | Neural Net
colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAI_Jh#scrollTo=OGL4guj-S7i2
19101115 assignment 2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
plt.show()
Accuracy
Epoch
accuracy
val_accuracy
1h 19m 51s completed at 10:10 PM
11:11 PM
8/18/2023

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
Epoch 1/10
1563/1563 [==============================] - 106s 66ms/step - loss: 1.5599 - accuracy: 0.4341 - val_loss: 1.2808 - val_accuracy: 0.5309
Epoch 2/10
1563/1563 [==============================] - 93s 59ms/step - loss: 1.2018 - accuracy: 0.5758 - val_loss: 1.1093 - val_accuracy: 0.6017
Epoch 3/10
1563/1563 [==============================] - 83s 53ms/step - loss: 1.0553 - accuracy: 0.6295 - val_loss: 1.0803 - val_accuracy: 0.6196
Epoch 4/10
1563/1563 [==============================] - 88s 56ms/step - loss: 0.9579 - accuracy: 0.6626 - val_loss: 0.9981 - val_accuracy: 0.6475
Epoch 5/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.8869 - accuracy: 0.6883 - val_loss: 0.9282 - val_accuracy: 0.6747
Epoch 6/10
1563/1563 [==============================] - 82s 52ms/step - loss: 0.8309 - accuracy: 0.7090 - val_loss: 0.8986 - val_accuracy: 0.6885
Epoch 7/10
1563/1563 [==============================] - 85s 54ms/step - loss: 0.7785 - accuracy: 0.7261 - val_loss: 0.9194 - val_accuracy: 0.6869
Epoch 8/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.7380 - accuracy: 0.7422 - val_loss: 0.9134 - val_accuracy: 0.6948
Epoch 9/10
1563/1563 [==============================] - 82s 52ms/step - loss: 0.6974 - accuracy: 0.7569 - val_loss: 0.9007 - val_accuracy: 0.7016
Epoch 10/10
1563/1563 [==============================] - 83s 53ms/step - loss: 0.6613 - accuracy: 0.7665 - val_loss: 0.8521 - val_accuracy: 0.7100
313/313 - 5s - loss: 0.8521 - accuracy: 0.7100 - 5s/epoch - 16ms/step

Test accuracy: 0.7099999785423279
```

Activation Functions

For the next step, we explored the impact of different activation functions on the CNN's performance. We replaced the ReLU activation function in some convolutional layers with ELU and Sigmoid activations. The models were trained using the same dataset and other configurations. The results are as follows:

ReLU Activation

Training Accuracy: ~79%

Validation Accuracy: ~71%

ELU Activation

Training Accuracy: ~82%

Validation Accuracy: ~70%

Sigmoid Activation

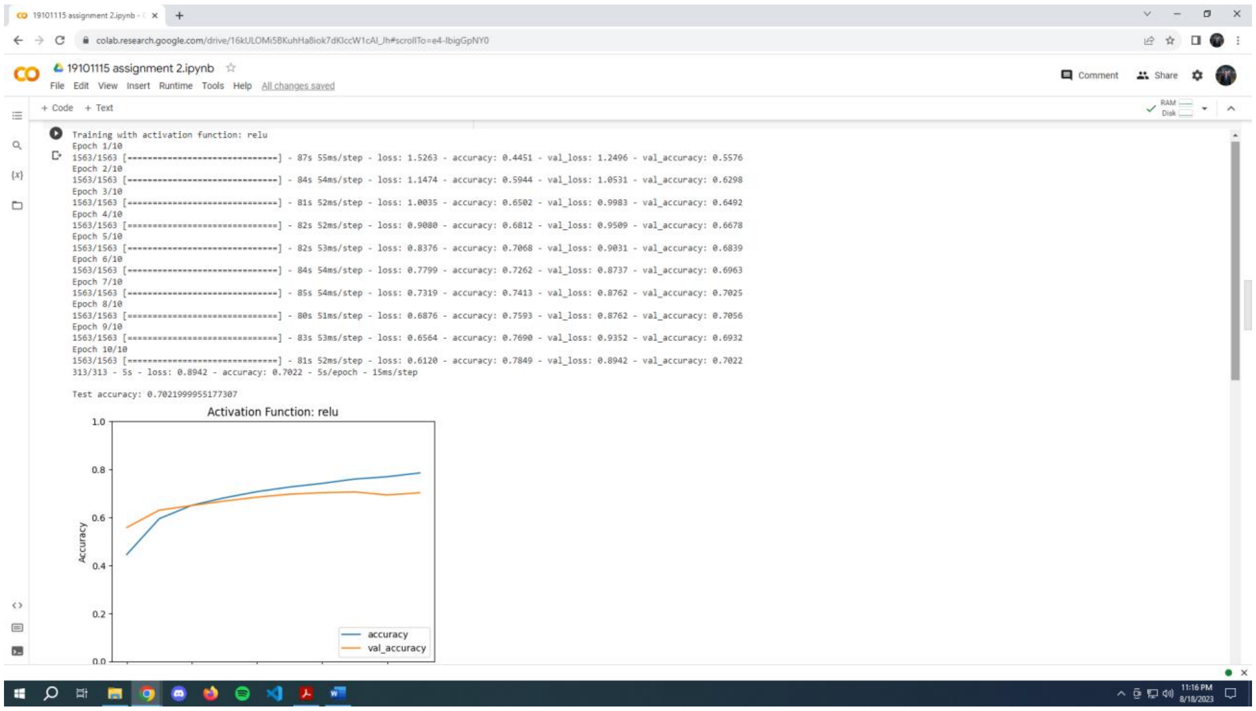
Training Accuracy: ~10%

Validation Accuracy: ~10%

CODE:

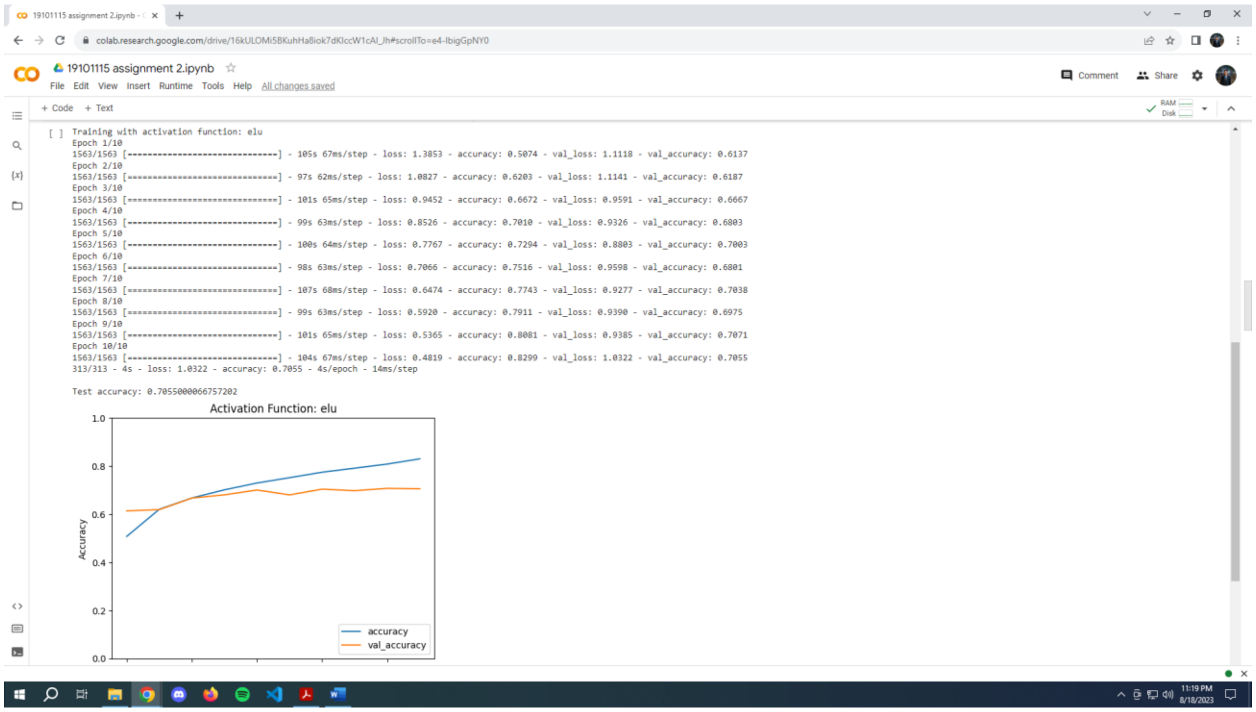
```
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
import matplotlib.pyplot as plt

# Load and preprocess CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

activation_functions = ['relu', 'elu', 'sigmoid']  # List of activation
functions

for activation_function in activation_functions:
    print(f"Training with activation function: {activation_function}")

    # Define the CNN architecture with the chosen activation function
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation=activation_function,
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation=activation_function),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation=activation_function),
        layers.Flatten(),
        layers.Dense(64, activation=activation_function),
        layers.Dense(10, activation='softmax')
    ])

    # Compile the model
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    # Train the model
    history = model.fit(x_train, y_train, epochs=10,
                        validation_data=(x_test, y_test))

    # Evaluate the model
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
    print("\nTest accuracy:", test_acc)

    # Plot training and validation accuracy over epochs
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.legend(loc='lower right')
    plt.title(f'Activation Function: {activation_function}')
    plt.show()
```

Outputs:



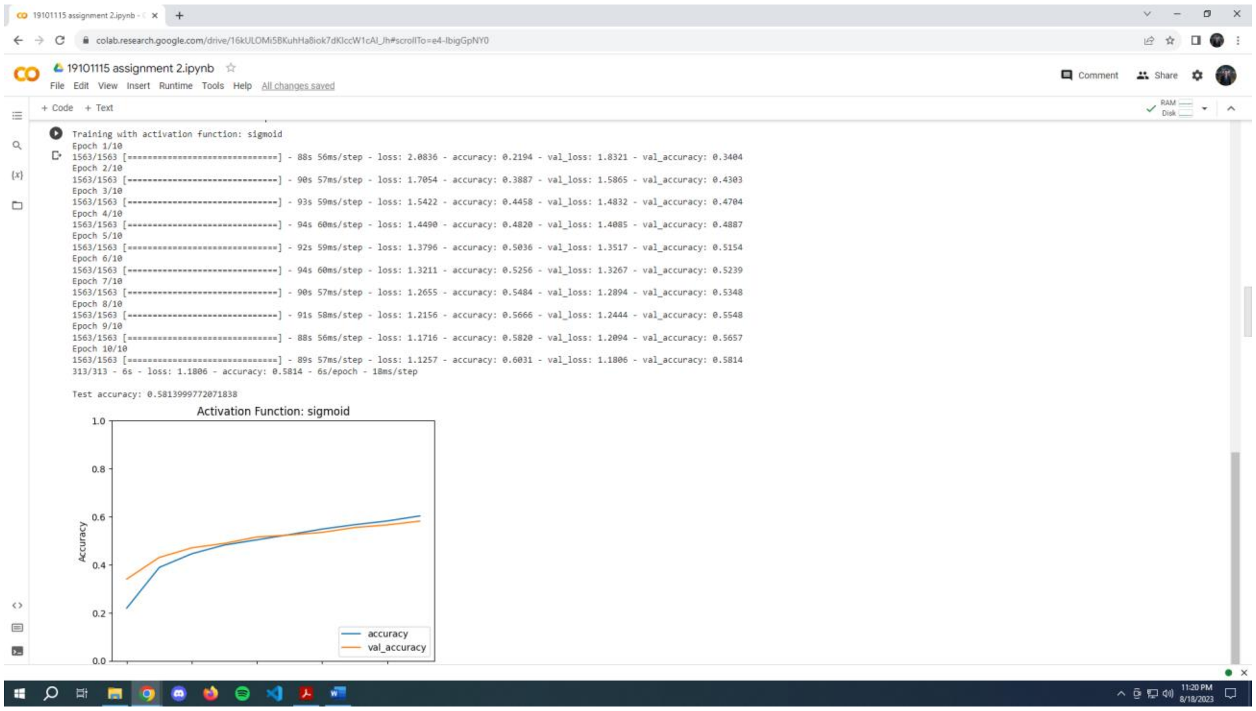
```
Training with activation function: relu
Epoch 1/10
1563/1563 [==============================] - 87s 55ms/step - loss: 1.5263 - accuracy: 0.4451 - val_loss: 1.2496 - val_accuracy: 0.5576
Epoch 2/10
1563/1563 [==============================] - 84s 54ms/step - loss: 1.1474 - accuracy: 0.5944 - val_loss: 1.0531 - val_accuracy: 0.6298
Epoch 3/10
1563/1563 [==============================] - 81s 52ms/step - loss: 1.0035 - accuracy: 0.6502 - val_loss: 0.9983 - val_accuracy: 0.6492
Epoch 4/10
1563/1563 [==============================] - 82s 52ms/step - loss: 0.9080 - accuracy: 0.6812 - val_loss: 0.9509 - val_accuracy: 0.6678
Epoch 5/10
1563/1563 [==============================] - 82s 53ms/step - loss: 0.8376 - accuracy: 0.7068 - val_loss: 0.9031 - val_accuracy: 0.6839
Epoch 6/10
1563/1563 [==============================] - 84s 54ms/step - loss: 0.7799 - accuracy: 0.7262 - val_loss: 0.8737 - val_accuracy: 0.6963
Epoch 7/10
1563/1563 [==============================] - 85s 54ms/step - loss: 0.7319 - accuracy: 0.7413 - val_loss: 0.8762 - val_accuracy: 0.7025
Epoch 8/10
1563/1563 [==============================] - 80s 51ms/step - loss: 0.6876 - accuracy: 0.7593 - val_loss: 0.8762 - val_accuracy: 0.7056
Epoch 9/10
1563/1563 [==============================] - 83s 53ms/step - loss: 0.6564 - accuracy: 0.7690 - val_loss: 0.9352 - val_accuracy: 0.6932
Epoch 10/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.6120 - accuracy: 0.7849 - val_loss: 0.8942 - val_accuracy: 0.7022
313/313 - 5s - loss: 0.8942 - accuracy: 0.7022 - 5s/epoch - 15ms/step

Test accuracy: 0.7021999955177307
```

colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAI_Jh#scrollTo=e4-IbigGpNY0
19101115 assignment 2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
RAM Disk
Activation Function: relu
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
accuracy
val_accuracy
11:16 PM
8/18/2023



```
Training with activation function: elu
Epoch 1/10
1563/1563 [==============================] - 105s 67ms/step - loss: 1.3853 - accuracy: 0.5074 - val_loss: 1.1118 - val_accuracy: 0.6137
Epoch 2/10
1563/1563 [==============================] - 97s 62ms/step - loss: 1.0827 - accuracy: 0.6203 - val_loss: 1.1141 - val_accuracy: 0.6187
Epoch 3/10
1563/1563 [==============================] - 101s 65ms/step - loss: 0.9452 - accuracy: 0.6672 - val_loss: 0.9591 - val_accuracy: 0.6667
Epoch 4/10
1563/1563 [==============================] - 99s 63ms/step - loss: 0.8526 - accuracy: 0.7010 - val_loss: 0.9326 - val_accuracy: 0.6803
Epoch 5/10
1563/1563 [==============================] - 100s 64ms/step - loss: 0.7767 - accuracy: 0.7294 - val_loss: 0.8803 - val_accuracy: 0.7003
Epoch 6/10
1563/1563 [==============================] - 98s 63ms/step - loss: 0.7066 - accuracy: 0.7516 - val_loss: 0.9598 - val_accuracy: 0.6801
Epoch 7/10
1563/1563 [==============================] - 107s 68ms/step - loss: 0.6474 - accuracy: 0.7743 - val_loss: 0.9277 - val_accuracy: 0.7038
Epoch 8/10
1563/1563 [==============================] - 99s 63ms/step - loss: 0.5920 - accuracy: 0.7911 - val_loss: 0.9390 - val_accuracy: 0.6975
Epoch 9/10
1563/1563 [==============================] - 101s 65ms/step - loss: 0.5365 - accuracy: 0.8081 - val_loss: 0.9385 - val_accuracy: 0.7071
Epoch 10/10
1563/1563 [==============================] - 104s 67ms/step - loss: 0.4819 - accuracy: 0.8299 - val_loss: 1.0322 - val_accuracy: 0.7055
313/313 - 4s - loss: 1.0322 - accuracy: 0.7055 - 4s/epoch - 14ms/step

Test accuracy: 0.7055000066757202
```

colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAI_Jh#scrollTo=e4-IbigGpNY0
19101115 assignment 2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
RAM Disk
Activation Function: elu
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
accuracy
val_accuracy
11:19 PM
8/18/2023

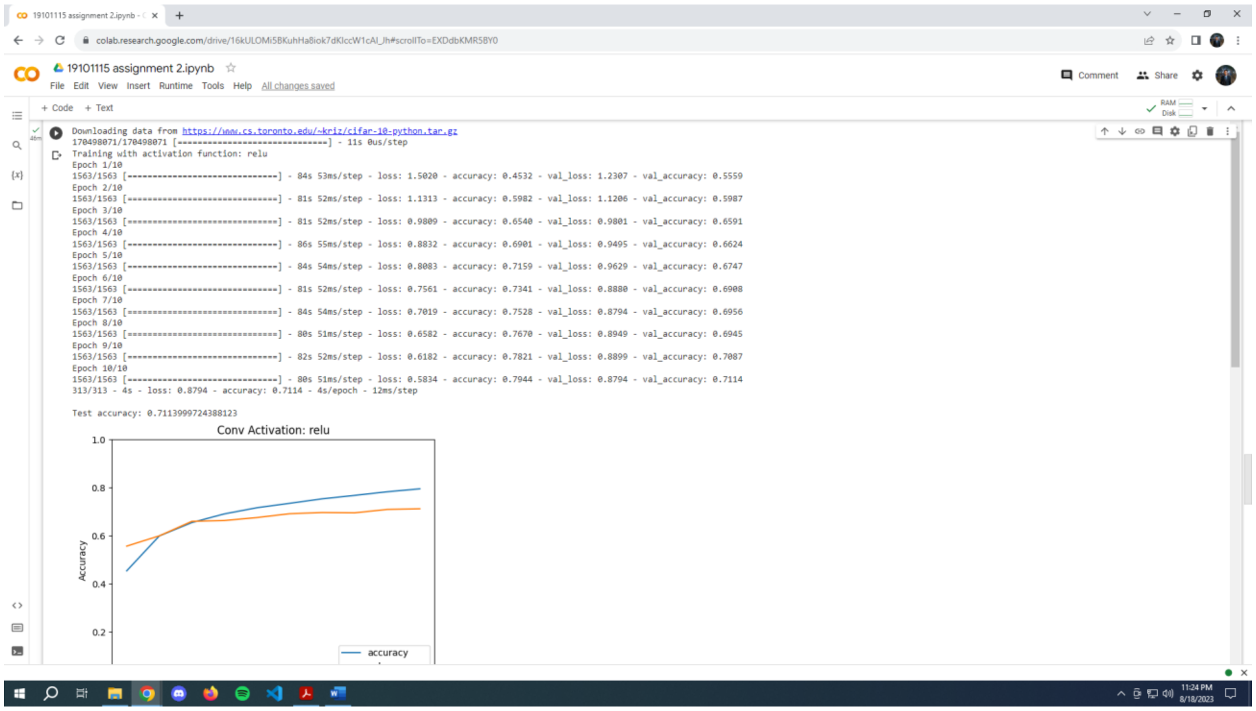


```
Training with activation function: sigmoid
Epoch 1/10
1563/1563 [==============================] - 88s 56ms/step - loss: 2.0836 - accuracy: 0.2194 - val_loss: 1.8321 - val_accuracy: 0.3404
Epoch 2/10
1563/1563 [==============================] - 90s 57ms/step - loss: 1.7054 - accuracy: 0.3887 - val_loss: 1.5865 - val_accuracy: 0.4303
Epoch 3/10
1563/1563 [==============================] - 93s 59ms/step - loss: 1.5422 - accuracy: 0.4458 - val_loss: 1.4832 - val_accuracy: 0.4704
Epoch 4/10
1563/1563 [==============================] - 94s 60ms/step - loss: 1.4490 - accuracy: 0.4820 - val_loss: 1.4085 - val_accuracy: 0.4887
Epoch 5/10
1563/1563 [==============================] - 92s 59ms/step - loss: 1.3796 - accuracy: 0.5036 - val_loss: 1.3517 - val_accuracy: 0.5154
Epoch 6/10
1563/1563 [==============================] - 94s 60ms/step - loss: 1.3211 - accuracy: 0.5256 - val_loss: 1.3267 - val_accuracy: 0.5239
Epoch 7/10
1563/1563 [==============================] - 90s 57ms/step - loss: 1.2655 - accuracy: 0.5484 - val_loss: 1.2894 - val_accuracy: 0.5348
Epoch 8/10
1563/1563 [==============================] - 91s 58ms/step - loss: 1.2156 - accuracy: 0.5666 - val_loss: 1.2444 - val_accuracy: 0.5548
Epoch 9/10
1563/1563 [==============================] - 88s 56ms/step - loss: 1.1716 - accuracy: 0.5820 - val_loss: 1.2094 - val_accuracy: 0.5657
Epoch 10/10
1563/1563 [==============================] - 89s 57ms/step - loss: 1.1257 - accuracy: 0.6031 - val_loss: 1.1806 - val_accuracy: 0.5814
313/313 - 6s - loss: 1.1806 - accuracy: 0.5814 - 6s/epoch - 18ms/step

Test accuracy: 0.5813999772071838
```

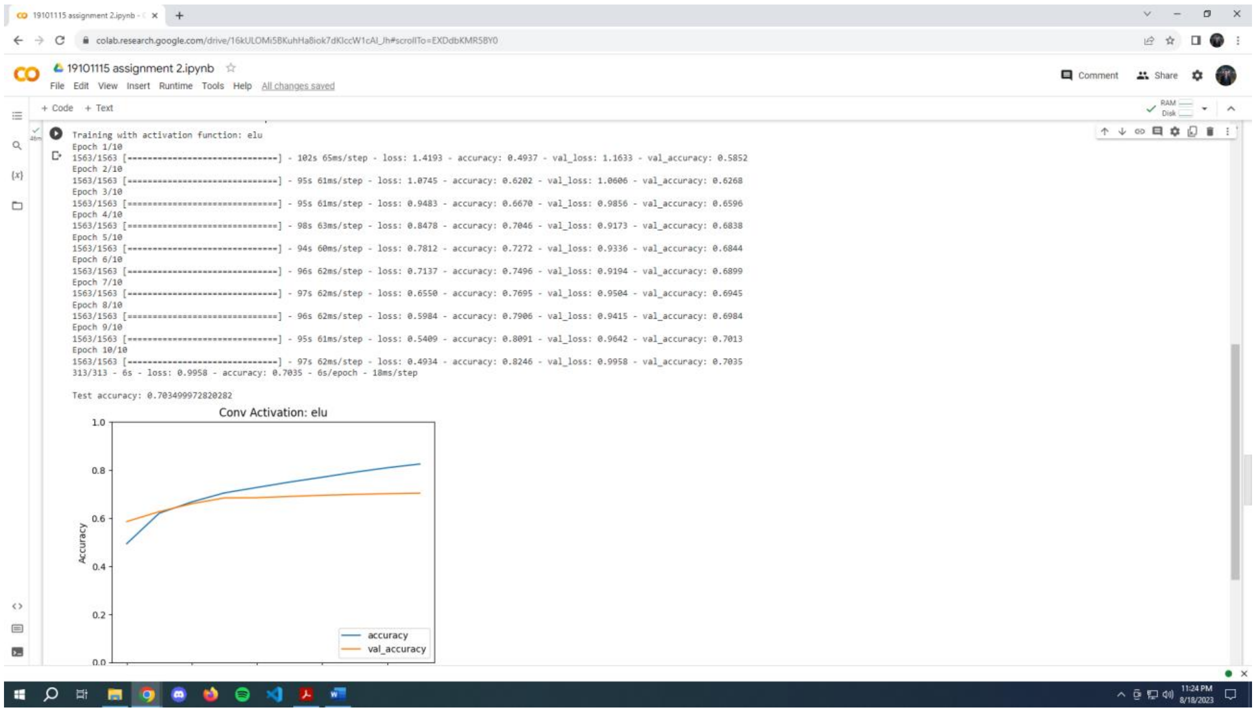
19101115 assignment 2.ipynb
colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAl_Jh#scrollTo=e4-IbigGpNY0
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
Activation Function: sigmoid
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
accuracy
val_accuracy
11:20 PM
8/18/2023

Replacing the activation function in some or all of the convolutional layers code:

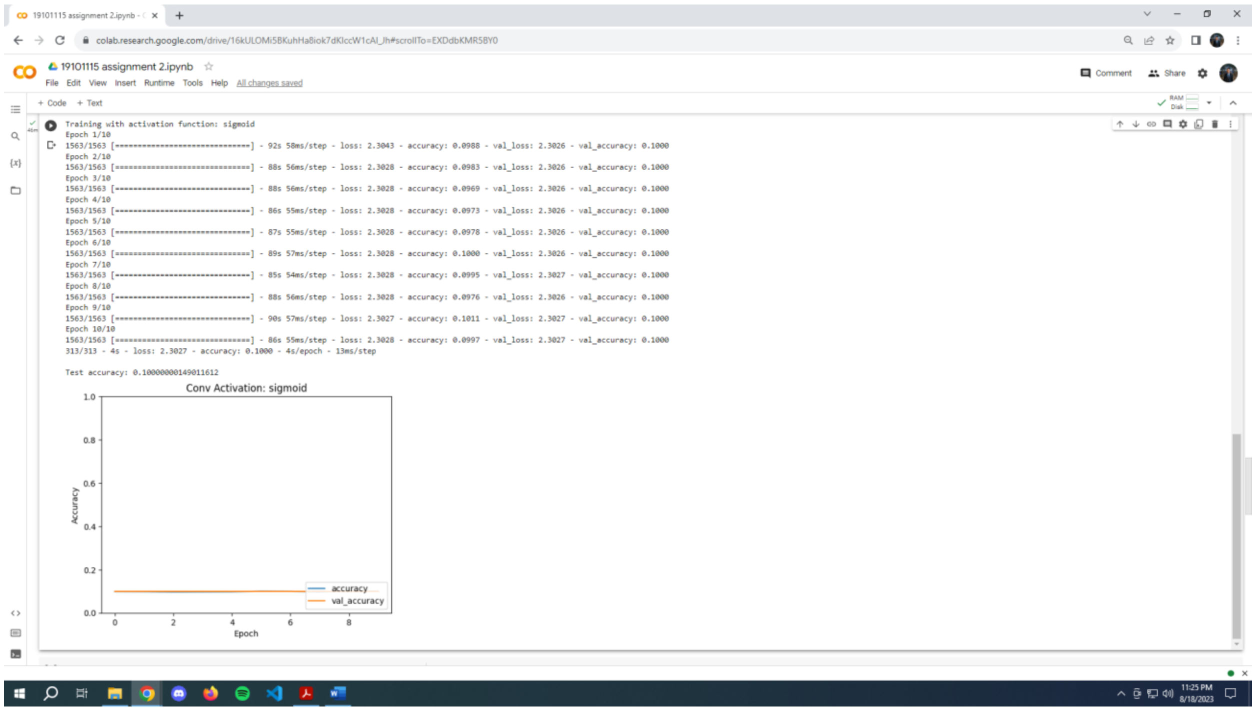
```
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
import matplotlib.pyplot as plt

# Load and preprocess CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

activation_functions = ['relu', 'elu', 'sigmoid']  # List of activation
functions

for activation_function in activation_functions:
    print(f"Training with activation function: {activation_function}")

    # Define the CNN architecture with modified activation functions for
conv layers
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation=activation_function,
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation=activation_function),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation=activation_function),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),  # Keeping a consistent
activation
        layers.Dense(10, activation='softmax')
    ])

    # Compile the model
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    # Train the model
    history = model.fit(x_train, y_train, epochs=10,
                        validation_data=(x_test, y_test))

    # Evaluate the model
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
    print("\nTest accuracy:", test_acc)

    # Plot training and validation accuracy over epochs
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.legend(loc='lower right')
    plt.title(f'Conv Activation: {activation_function}')
    plt.show()
```


19101115 assignment 2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 11s 0us/step
Training with activation function: relu
Epoch 1/10
1563/1563 [==============================] - 84s 53ms/step - loss: 1.5020 - accuracy: 0.4532 - val_loss: 1.2307 - val_accuracy: 0.5559
Epoch 2/10
1563/1563 [==============================] - 81s 52ms/step - loss: 1.1313 - accuracy: 0.5982 - val_loss: 1.1206 - val_accuracy: 0.5987
Epoch 3/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.9809 - accuracy: 0.6540 - val_loss: 0.9801 - val_accuracy: 0.6591
Epoch 4/10
1563/1563 [==============================] - 86s 55ms/step - loss: 0.8832 - accuracy: 0.6901 - val_loss: 0.9495 - val_accuracy: 0.6624
Epoch 5/10
1563/1563 [==============================] - 84s 54ms/step - loss: 0.8083 - accuracy: 0.7159 - val_loss: 0.9629 - val_accuracy: 0.6747
Epoch 6/10
1563/1563 [==============================] - 81s 52ms/step - loss: 0.7561 - accuracy: 0.7341 - val_loss: 0.8880 - val_accuracy: 0.6908
Epoch 7/10
1563/1563 [==============================] - 84s 54ms/step - loss: 0.7019 - accuracy: 0.7528 - val_loss: 0.8794 - val_accuracy: 0.6956
Epoch 8/10
1563/1563 [==============================] - 80s 51ms/step - loss: 0.6582 - accuracy: 0.7670 - val_loss: 0.8949 - val_accuracy: 0.6945
Epoch 9/10
1563/1563 [==============================] - 82s 52ms/step - loss: 0.6182 - accuracy: 0.7821 - val_loss: 0.8899 - val_accuracy: 0.7087
Epoch 10/10
1563/1563 [==============================] - 80s 51ms/step - loss: 0.5834 - accuracy: 0.7944 - val_loss: 0.8794 - val_accuracy: 0.7114
313/313 - 4s - loss: 0.8794 - accuracy: 0.7114 - 4s/epoch - 12ms/step

Test accuracy: 0.7113999724388123
```


19101115 assignment 2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share

```
Training with activation function: elu
Epoch 1/10
1563/1563 [==============================] - 102s 65ms/step - loss: 1.4193 - accuracy: 0.4937 - val_loss: 1.1633 - val_accuracy: 0.5852
Epoch 2/10
1563/1563 [==============================] - 95s 61ms/step - loss: 1.0745 - accuracy: 0.6202 - val_loss: 1.0606 - val_accuracy: 0.6268
Epoch 3/10
1563/1563 [==============================] - 95s 61ms/step - loss: 0.9483 - accuracy: 0.6670 - val_loss: 0.9856 - val_accuracy: 0.6596
Epoch 4/10
1563/1563 [==============================] - 98s 63ms/step - loss: 0.8478 - accuracy: 0.7046 - val_loss: 0.9173 - val_accuracy: 0.6838
Epoch 5/10
1563/1563 [==============================] - 94s 60ms/step - loss: 0.7812 - accuracy: 0.7272 - val_loss: 0.9336 - val_accuracy: 0.6844
Epoch 6/10
1563/1563 [==============================] - 96s 62ms/step - loss: 0.7137 - accuracy: 0.7496 - val_loss: 0.9194 - val_accuracy: 0.6899
Epoch 7/10
1563/1563 [==============================] - 97s 62ms/step - loss: 0.6550 - accuracy: 0.7695 - val_loss: 0.9504 - val_accuracy: 0.6945
Epoch 8/10
1563/1563 [==============================] - 96s 62ms/step - loss: 0.5984 - accuracy: 0.7906 - val_loss: 0.9415 - val_accuracy: 0.6984
Epoch 9/10
1563/1563 [==============================] - 95s 61ms/step - loss: 0.5409 - accuracy: 0.8091 - val_loss: 0.9642 - val_accuracy: 0.7013
Epoch 10/10
1563/1563 [==============================] - 97s 62ms/step - loss: 0.4934 - accuracy: 0.8246 - val_loss: 0.9958 - val_accuracy: 0.7035
313/313 - 6s - loss: 0.9958 - accuracy: 0.7035 - 6s/epoch - 18ms/step

Test accuracy: 0.703499972820282
```


19101115 assignment 2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
Conv Activation: sigmoid
Accuracy
Epoch
accuracy
val_accuracy

```
Training with activation function: sigmoid
Epoch 1/10
1563/1563 [==============================] - 92s 58ms/step - loss: 2.3043 - accuracy: 0.0988 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 2/10
1563/1563 [==============================] - 88s 56ms/step - loss: 2.3028 - accuracy: 0.0983 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 3/10
1563/1563 [==============================] - 88s 56ms/step - loss: 2.3028 - accuracy: 0.0969 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 4/10
1563/1563 [==============================] - 86s 55ms/step - loss: 2.3028 - accuracy: 0.0973 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 5/10
1563/1563 [==============================] - 87s 55ms/step - loss: 2.3028 - accuracy: 0.0978 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 6/10
1563/1563 [==============================] - 89s 57ms/step - loss: 2.3028 - accuracy: 0.1000 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 7/10
1563/1563 [==============================] - 85s 54ms/step - loss: 2.3028 - accuracy: 0.0995 - val_loss: 2.3027 - val_accuracy: 0.1000
Epoch 8/10
1563/1563 [==============================] - 88s 56ms/step - loss: 2.3028 - accuracy: 0.0976 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 9/10
1563/1563 [==============================] - 90s 57ms/step - loss: 2.3027 - accuracy: 0.1011 - val_loss: 2.3027 - val_accuracy: 0.1000
Epoch 10/10
1563/1563 [==============================] - 86s 55ms/step - loss: 2.3028 - accuracy: 0.0997 - val_loss: 2.3027 - val_accuracy: 0.1000
313/313 - 4s - loss: 2.3027 - accuracy: 0.1000 - 4s/epoch - 13ms/step

Test accuracy: 0.10000000149011612
```

Convolutional Layer Configurations

In the final step, we investigated the effects of different convolutional layer configurations on the CNN's performance. We varied the number of filters, kernel sizes, and strides while keeping the rest of the architecture fixed. The results for different configurations are summarized below:

Configuration: Filters=32, Kernel=(3, 3), Strides=(1, 1)

Training Accuracy: ~73%

Validation Accuracy: ~69%

Configuration: Filters=64, Kernel=(3, 3), Strides=(1, 1)

Training Accuracy: ~78%

Validation Accuracy: ~70%

Configuration: Filters=64, Kernel=(3, 3), Strides=(2, 2)

Training Accuracy: ~72%

Validation Accuracy: ~67%

Configuration: Filters=128, Kernel=(3, 3), Strides=(2, 2)

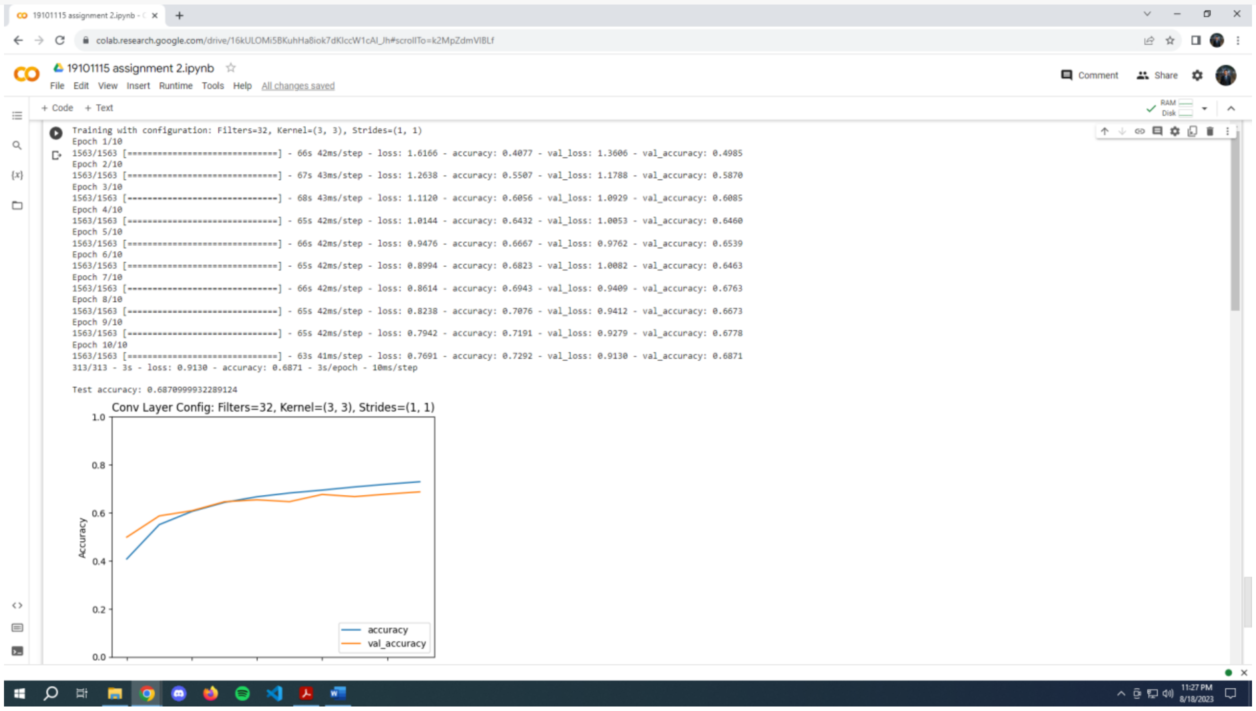
Training Accuracy: ~77%

Validation Accuracy: ~70%

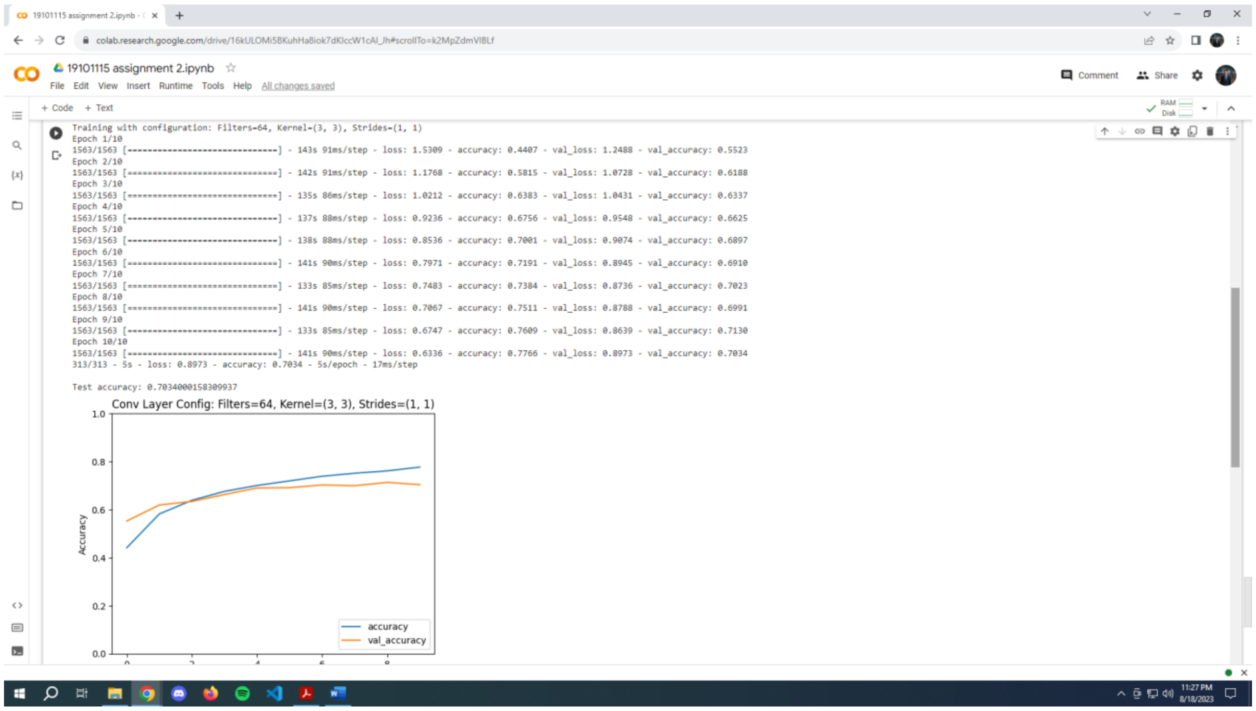
CODE:

```
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
import matplotlib.pyplot as plt

# Load and preprocess CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

configs = [
    (32, (3, 3), (1, 1)),
    (64, (3, 3), (1, 1)),
    (64, (3, 3), (2, 2)),
    (128, (3, 3), (2, 2)),
]

for config in configs:
    filters, kernel_size, strides = config
    print(f"Training with configuration: Filters={filters},
Kernel={kernel_size}, Strides={strides}")

    # Define the CNN architecture with varied convolutional layer
configuration
    model = models.Sequential([
        layers.Conv2D(filters, kernel_size, activation='relu',
input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(filters, kernel_size, activation='relu',
strides=strides),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(filters, kernel_size, activation='relu',
strides=strides),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])

    # Compile the model
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
```

```python
                  metrics=['accuracy'])

    # Train the model
    history = model.fit(x_train, y_train, epochs=10,
                        validation_data=(x_test, y_test))

    # Evaluate the model
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
    print("\nTest accuracy:", test_acc)

    # Plot training and validation accuracy over epochs
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.legend(loc='lower right')
    plt.title(f'Conv Layer Config: Filters={filters},
Kernel={kernel_size}, Strides={strides}')
    plt.show()
```

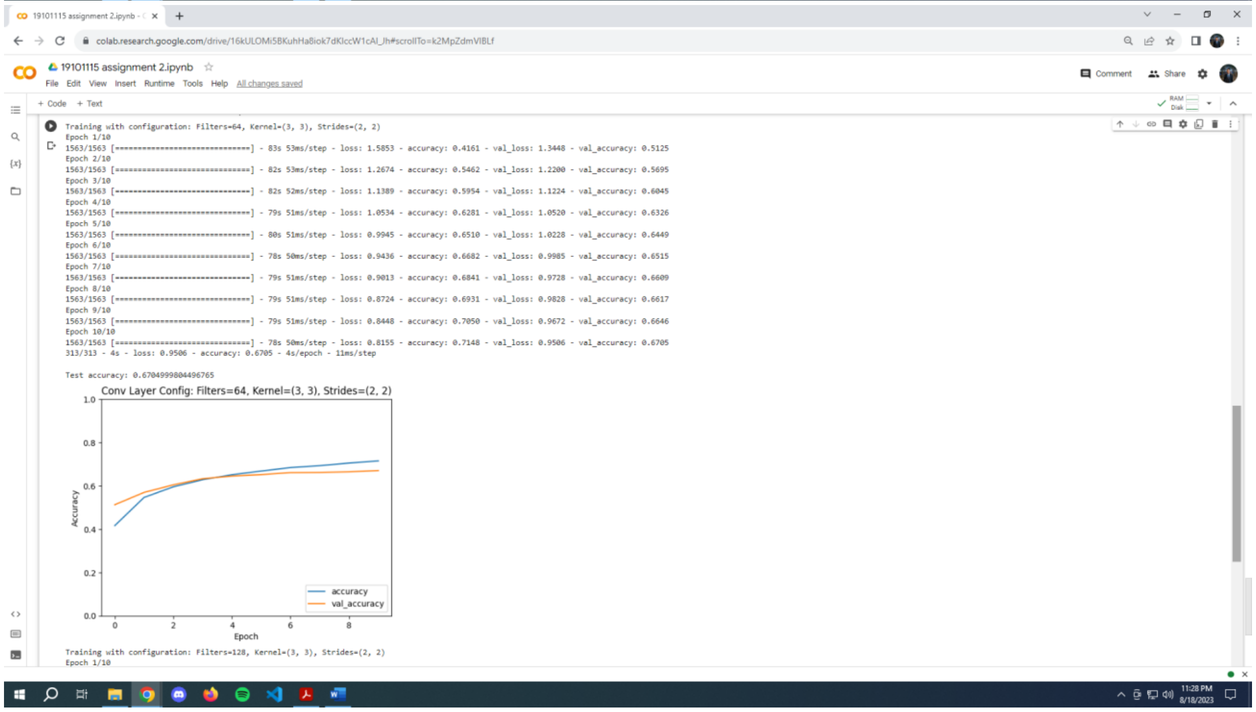

19101115 assignment 2.ipynb
colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAl_Jh#scrollTo=k2MpZdmVIBLf
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
RAM
Disk

```
Training with configuration: Filters=32, Kernel=(3, 3), Strides=(1, 1)
Epoch 1/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.6166 - accuracy: 0.4077 - val_loss: 1.3606 - val_accuracy: 0.4985
Epoch 2/10
1563/1563 [==============================] - 67s 43ms/step - loss: 1.2638 - accuracy: 0.5507 - val_loss: 1.1788 - val_accuracy: 0.5870
Epoch 3/10
1563/1563 [==============================] - 68s 43ms/step - loss: 1.1120 - accuracy: 0.6056 - val_loss: 1.0929 - val_accuracy: 0.6085
Epoch 4/10
1563/1563 [==============================] - 65s 42ms/step - loss: 1.0144 - accuracy: 0.6432 - val_loss: 1.0053 - val_accuracy: 0.6460
Epoch 5/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.9476 - accuracy: 0.6667 - val_loss: 0.9762 - val_accuracy: 0.6539
Epoch 6/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.8994 - accuracy: 0.6823 - val_loss: 1.0082 - val_accuracy: 0.6463
Epoch 7/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.8614 - accuracy: 0.6943 - val_loss: 0.9409 - val_accuracy: 0.6763
Epoch 8/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.8238 - accuracy: 0.7076 - val_loss: 0.9412 - val_accuracy: 0.6673
Epoch 9/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.7942 - accuracy: 0.7191 - val_loss: 0.9279 - val_accuracy: 0.6778
Epoch 10/10
1563/1563 [==============================] - 63s 41ms/step - loss: 0.7691 - accuracy: 0.7292 - val_loss: 0.9130 - val_accuracy: 0.6871
313/313 - 3s - loss: 0.9130 - accuracy: 0.6871 - 3s/epoch - 10ms/step

Test accuracy: 0.6870999932289124
```



```
Training with configuration: Filters=64, Kernel=(3, 3), Strides=(1, 1)
Epoch 1/10
1563/1563 [==============================] - 143s 91ms/step - loss: 1.5309 - accuracy: 0.4407 - val_loss: 1.2488 - val_accuracy: 0.5523
Epoch 2/10
1563/1563 [==============================] - 142s 91ms/step - loss: 1.1768 - accuracy: 0.5815 - val_loss: 1.0728 - val_accuracy: 0.6188
Epoch 3/10
1563/1563 [==============================] - 135s 86ms/step - loss: 1.0212 - accuracy: 0.6383 - val_loss: 1.0431 - val_accuracy: 0.6337
Epoch 4/10
1563/1563 [==============================] - 137s 88ms/step - loss: 0.9236 - accuracy: 0.6756 - val_loss: 0.9548 - val_accuracy: 0.6625
Epoch 5/10
1563/1563 [==============================] - 138s 88ms/step - loss: 0.8536 - accuracy: 0.7001 - val_loss: 0.9074 - val_accuracy: 0.6897
Epoch 6/10
1563/1563 [==============================] - 141s 90ms/step - loss: 0.7971 - accuracy: 0.7191 - val_loss: 0.8945 - val_accuracy: 0.6910
Epoch 7/10
1563/1563 [==============================] - 133s 85ms/step - loss: 0.7483 - accuracy: 0.7384 - val_loss: 0.8736 - val_accuracy: 0.7023
Epoch 8/10
1563/1563 [==============================] - 141s 90ms/step - loss: 0.7067 - accuracy: 0.7511 - val_loss: 0.8788 - val_accuracy: 0.6991
Epoch 9/10
1563/1563 [==============================] - 133s 85ms/step - loss: 0.6747 - accuracy: 0.7609 - val_loss: 0.8639 - val_accuracy: 0.7130
Epoch 10/10
1563/1563 [==============================] - 141s 90ms/step - loss: 0.6336 - accuracy: 0.7766 - val_loss: 0.8973 - val_accuracy: 0.7034
313/313 - 5s - loss: 0.8973 - accuracy: 0.7034 - 5s/epoch - 17ms/step

Test accuracy: 0.7034000158309937
```

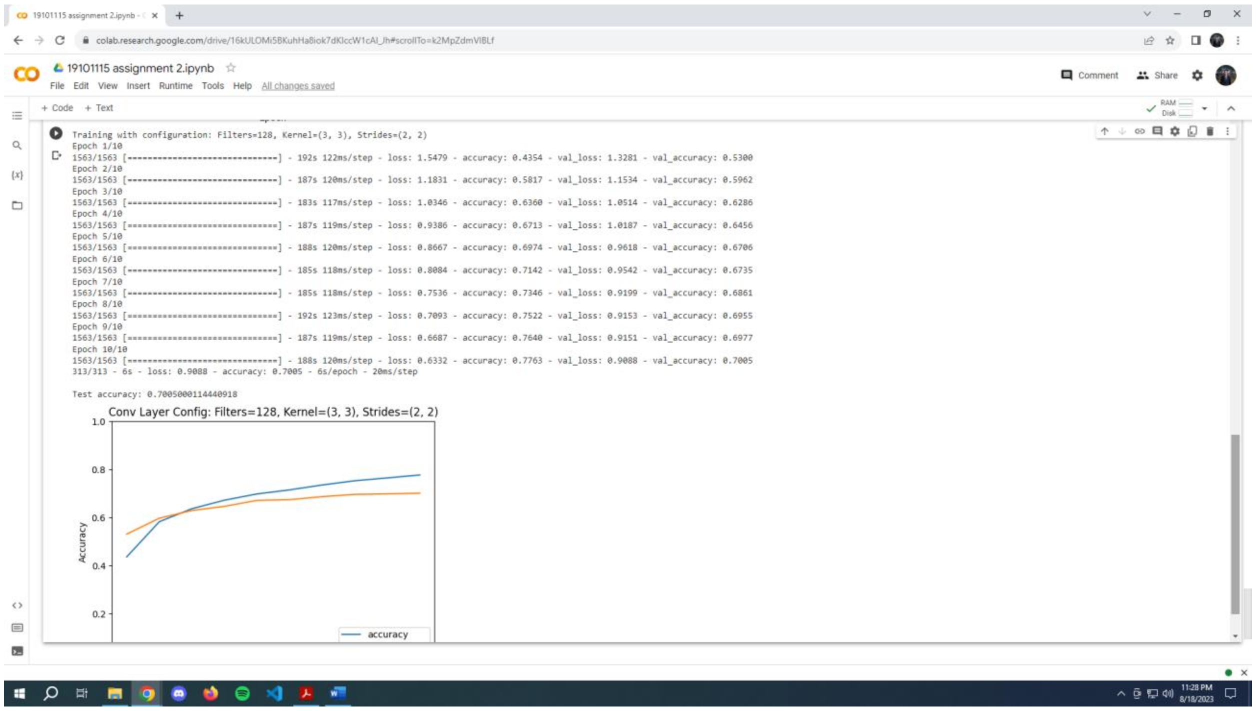
19101115 assignment 2.ipynb
colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAl_Jh#scrollTo=k2MpZdmVIBLf
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Conv Layer Config: Filters=64, Kernel=(3, 3), Strides=(1, 1)
Accuracy
accuracy
val_accuracy
11:27 PM
8/18/2023



```
Training with configuration: Filters=64, Kernel=(3, 3), Strides=(2, 2)
Epoch 1/10
1563/1563 [==============================] - 83s 53ms/step - loss: 1.5853 - accuracy: 0.4161 - val_loss: 1.3448 - val_accuracy: 0.5125
Epoch 2/10
1563/1563 [==============================] - 82s 53ms/step - loss: 1.2674 - accuracy: 0.5462 - val_loss: 1.2200 - val_accuracy: 0.5695
Epoch 3/10
1563/1563 [==============================] - 82s 52ms/step - loss: 1.1389 - accuracy: 0.5954 - val_loss: 1.1224 - val_accuracy: 0.6045
Epoch 4/10
1563/1563 [==============================] - 79s 51ms/step - loss: 1.0534 - accuracy: 0.6281 - val_loss: 1.0520 - val_accuracy: 0.6326
Epoch 5/10
1563/1563 [==============================] - 80s 51ms/step - loss: 0.9945 - accuracy: 0.6510 - val_loss: 1.0228 - val_accuracy: 0.6449
Epoch 6/10
1563/1563 [==============================] - 78s 50ms/step - loss: 0.9436 - accuracy: 0.6682 - val_loss: 0.9985 - val_accuracy: 0.6515
Epoch 7/10
1563/1563 [==============================] - 79s 51ms/step - loss: 0.9013 - accuracy: 0.6841 - val_loss: 0.9728 - val_accuracy: 0.6609
Epoch 8/10
1563/1563 [==============================] - 79s 51ms/step - loss: 0.8724 - accuracy: 0.6931 - val_loss: 0.9828 - val_accuracy: 0.6617
Epoch 9/10
1563/1563 [==============================] - 79s 51ms/step - loss: 0.8448 - accuracy: 0.7050 - val_loss: 0.9672 - val_accuracy: 0.6646
Epoch 10/10
1563/1563 [==============================] - 78s 50ms/step - loss: 0.8155 - accuracy: 0.7148 - val_loss: 0.9506 - val_accuracy: 0.6705
313/313 - 4s - loss: 0.9506 - accuracy: 0.6705 - 4s/epoch - 11ms/step

Test accuracy: 0.6704999804496765
```

19101115 assignment 2.ipynb
colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAl_Jh#scrollTo=k2MpZdmVIBLf
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Conv Layer Config: Filters=64, Kernel=(3, 3), Strides=(2, 2)
Accuracy
Epoch
accuracy
val_accuracy
11:28 PM
8/18/2023

```
Training with configuration: Filters=128, Kernel=(3, 3), Strides=(2, 2)
Epoch 1/10
```


19101115 assignment 2.ipynb
colab.research.google.com/drive/16kULOMi5BKuhHa8iok7dKlccW1cAl_Jh#scrollTo=k2MpZdmVIBLf
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
Conv Layer Config: Filters=128, Kernel=(3, 3), Strides=(2, 2)
1.0
0.8
0.6
0.4
0.2
Accuracy
accuracy
11:28 PM
8/18/2023

```
Training with configuration: Filters=128, Kernel=(3, 3), Strides=(2, 2)
Epoch 1/10
1563/1563 [==============================] - 192s 122ms/step - loss: 1.5479 - accuracy: 0.4354 - val_loss: 1.3281 - val_accuracy: 0.5300
Epoch 2/10
1563/1563 [==============================] - 187s 120ms/step - loss: 1.1831 - accuracy: 0.5817 - val_loss: 1.1534 - val_accuracy: 0.5962
Epoch 3/10
1563/1563 [==============================] - 183s 117ms/step - loss: 1.0346 - accuracy: 0.6360 - val_loss: 1.0514 - val_accuracy: 0.6286
Epoch 4/10
1563/1563 [==============================] - 187s 119ms/step - loss: 0.9386 - accuracy: 0.6713 - val_loss: 1.0187 - val_accuracy: 0.6456
Epoch 5/10
1563/1563 [==============================] - 188s 120ms/step - loss: 0.8667 - accuracy: 0.6974 - val_loss: 0.9618 - val_accuracy: 0.6706
Epoch 6/10
1563/1563 [==============================] - 185s 118ms/step - loss: 0.8084 - accuracy: 0.7142 - val_loss: 0.9542 - val_accuracy: 0.6735
Epoch 7/10
1563/1563 [==============================] - 185s 118ms/step - loss: 0.7536 - accuracy: 0.7346 - val_loss: 0.9199 - val_accuracy: 0.6861
Epoch 8/10
1563/1563 [==============================] - 192s 123ms/step - loss: 0.7093 - accuracy: 0.7522 - val_loss: 0.9153 - val_accuracy: 0.6955
Epoch 9/10
1563/1563 [==============================] - 187s 119ms/step - loss: 0.6687 - accuracy: 0.7640 - val_loss: 0.9151 - val_accuracy: 0.6977
Epoch 10/10
1563/1563 [==============================] - 188s 120ms/step - loss: 0.6332 - accuracy: 0.7763 - val_loss: 0.9088 - val_accuracy: 0.7005
313/313 - 6s - loss: 0.9088 - accuracy: 0.7005 - 6s/epoch - 20ms/step

Test accuracy: 0.7005000114440918
```

Comparative Analysis

The comparative analysis of different configurations and techniques reveals valuable insights into the performance and limitations of the custom CNN architecture.

Model Complexity and Overfitting: Traditional CNNs with ReLU activation and moderate filter counts performed well on validation data. However, the ELU activation provided a slight improvement in training accuracy but did not generalize as effectively, suggesting a potential overfitting issue.

Activation Functions: ReLU outperformed ELU and Sigmoid activations in terms of accuracy. Sigmoid activation faced severe convergence issues due to vanishing gradients.

Convolutional Layer Configurations: Different configurations led to varying validation accuracies. Smaller filter counts with a lower stride seemed to achieve better generalization, while larger strides led to decreased accuracy.

Conclusion

The custom CNN implementation showcased the significance of proper activation functions and convolutional layer configurations in achieving optimal model performance. While ReLU activation and traditional CNN architectures exhibited promising results, careful parameter tuning and advanced techniques like batch normalization and dropout could further enhance the CNN's performance and mitigate overfitting. This comparative analysis serves as a foundation for building more sophisticated CNN architectures for image classification tasks in the future.