BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

1. What you learned in the ICP:

   In this ICP I learned how to build an image classification using Convolutional neural network model for the cifar10 dataset. Also, I worked on the data augmentation to prevent the overfitting.

2. ICP description what was the task you were performing:

   I performed image classification in the Cifar10 dataset using CNN model. I changed 4 hyper parameters in the model and will explain what are that in the following section. Also, I have uploaded 5 different images (out of the dataset) to validate the model.

3. Challenges that you faced:

   I could not use GPU runtime in light of I have reached the limit of usage. I got the following message from Google Colab (You cannot currently connect to a GPU due to usage limits in Colab). However, I solved this issue by using another Google account.

4. <u>Screen shots that shows the successful execution of each required step of your code:</u>

Hyper-parameter #1: (Changing the activation layer = Softmax):

I changed the activation function in output layer to “softmax” instead of “relu”, and I found that the model accuracy has decreased (Accuracy: 51.32%) and the loss was still high (1.3591). Also, I noticed that the model can’t predict correctly as already did while I was using “relu” activation function.

ICP5_BDAA.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[155] model = Sequential([
        data_augmentation,
        layers.experimental.preprocessing.Rescaling(1./255),
        layers.Conv2D(16, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(32, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Dropout(0.2),
        layers.Flatten(),
        layers.Dense(128, activation='softmax'),
        layers.Dense(class_num)
      ])
```

compile and train the model

```
[156] model.compile(optimizer='adam',
                    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                    metrics=['accuracy'])
```

ICP5_BDAA.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
epochs = 10
history = model.fit(
  X_train,
  y_train,
  validation_data=(X_test, y_test),
  epochs=epochs,
  batch_size=64,
  )
```

```
Epoch 1/10
782/782 [==============================] - 11s 12ms/step - loss: 2.1524 - accuracy: 0.2049 - val_loss: 2.0111 - val_accuracy: 0.2644
Epoch 2/10
782/782 [==============================] - 9s 11ms/step - loss: 1.9638 - accuracy: 0.2760 - val_loss: 1.8472 - val_accuracy: 0.3114
Epoch 3/10
782/782 [==============================] - 9s 11ms/step - loss: 1.8189 - accuracy: 0.3253 - val_loss: 1.7552 - val_accuracy: 0.3473
Epoch 4/10
782/782 [==============================] - 9s 11ms/step - loss: 1.6912 - accuracy: 0.3859 - val_loss: 1.5902 - val_accuracy: 0.4193
Epoch 5/10
782/782 [==============================] - 9s 11ms/step - loss: 1.5852 - accuracy: 0.4258 - val_loss: 1.4914 - val_accuracy: 0.4469
Epoch 6/10
782/782 [==============================] - 9s 11ms/step - loss: 1.5132 - accuracy: 0.4463 - val_loss: 1.5228 - val_accuracy: 0.4474
Epoch 7/10
782/782 [==============================] - 9s 11ms/step - loss: 1.4577 - accuracy: 0.4644 - val_loss: 1.3907 - val_accuracy: 0.4774
Epoch 8/10
782/782 [==============================] - 9s 11ms/step - loss: 1.4245 - accuracy: 0.4773 - val_loss: 1.3453 - val_accuracy: 0.4963
Epoch 9/10
782/782 [==============================] - 8s 11ms/step - loss: 1.3922 - accuracy: 0.4877 - val_loss: 1.3478 - val_accuracy: 0.5075
Epoch 10/10
782/782 [==============================] - 9s 11ms/step - loss: 1.3591 - accuracy: 0.5018 - val_loss: 1.2954 - val_accuracy: 0.5132
```

```
[159] #model as given has reached = Accuracy: 65.71%
      # Accuracy: 51.32% with softmax

      model_eval = model.evaluate(X_test, y_test, verbose=0)
      print("Accuracy: %.2f%%" % (model_eval[1]*100))
```

```
Accuracy: 51.32%
```

Hyper-parameter #2: (Conv2D filter 32, output dense layer activiation function = relu with 256):

I have increased the convolutional layer on 32 input channels and 256 output which means that every 32 channels was traversed by 256 3x3 kernels in result of 8,192 feature maps. In this change, I noticed that the model has performed much better than other changes. The accuracy is 73.02%.

ICP5_BDAA.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[179] model = Sequential([
        data_augmentation,
        layers.experimental.preprocessing.Rescaling(1./255),
        layers.Conv2D(32, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(128, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Dropout(0.2),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dense(class_num)
      ])
```

compile and train the model

```
[180] model.compile(optimizer='adam',
                    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                    metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_16 (Sequential)   (None, 32, 32, 3)         0
_________________________________________________________________
rescaling_13 (Rescaling)     (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 16, 16, 64)        18496
```

BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

ICP5_BDAA.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[182] epochs = 10
      history = model.fit(
        X_train,
        y_train,
        validation_data=(X_test, y_test),
        epochs=epochs,
        batch_size=64,
        )
```

```
Epoch 1/10
782/782 [==============================] - 15s 17ms/step - loss: 1.6194 - accuracy: 0.4104 - val_loss: 1.3020 - val_accuracy: 0.5303
Epoch 2/10
782/782 [==============================] - 12s 16ms/step - loss: 1.2889 - accuracy: 0.5390 - val_loss: 1.2697 - val_accuracy: 0.5674
Epoch 3/10
782/782 [==============================] - 12s 16ms/step - loss: 1.1638 - accuracy: 0.5873 - val_loss: 1.0741 - val_accuracy: 0.6251
Epoch 4/10
782/782 [==============================] - 12s 16ms/step - loss: 1.0909 - accuracy: 0.6163 - val_loss: 0.9704 - val_accuracy: 0.6626
Epoch 5/10
782/782 [==============================] - 13s 16ms/step - loss: 1.0302 - accuracy: 0.6346 - val_loss: 1.0373 - val_accuracy: 0.6438
Epoch 6/10
782/782 [==============================] - 12s 16ms/step - loss: 0.9843 - accuracy: 0.6515 - val_loss: 0.8733 - val_accuracy: 0.6996
Epoch 7/10
782/782 [==============================] - 12s 16ms/step - loss: 0.9422 - accuracy: 0.6656 - val_loss: 0.8940 - val_accuracy: 0.6886
Epoch 8/10
782/782 [==============================] - 13s 16ms/step - loss: 0.9108 - accuracy: 0.6805 - val_loss: 0.8443 - val_accuracy: 0.7040
Epoch 9/10
782/782 [==============================] - 13s 16ms/step - loss: 0.8844 - accuracy: 0.6892 - val_loss: 0.8492 - val_accuracy: 0.7026
Epoch 10/10
782/782 [==============================] - 13s 16ms/step - loss: 0.8604 - accuracy: 0.6970 - val_loss: 0.7971 - val_accuracy: 0.7302
```

```
#model as given has reached = Accuracy: 65.71%
# Accuracy: 51.32% with softmax
# Accuracy: 73.02% with relu activiation layer and conv2d start 32

model_eval = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (model_eval[1]*100))
```

```
Accuracy: 73.02%
```

Hyper-parameter #3: (Dropout = 0.7):

I have played with the Dropout many times in order to see what is going to happen to the model. In the first time, I have added the (Dropout 0.2) before the data augmentation, but I could not see any big different on the model performance nor the training loss. However, I removed the one before the data augmentation and increase the one after to be (0.7) and I found that the model is underfitted. The good thing is that I've prevent the model to be overfitting, but the model has underfitted. Therefore, the dropout layer rate should be a 0.2 to fit the model.

BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

ICP5_BDAA (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

randomly from the applied layer

```
[49] model = Sequential([
       data_augmentation,
       layers.experimental.preprocessing.Rescaling(1./255),
       layers.Conv2D(32, 3, padding='same', activation='relu'),
       layers.MaxPooling2D(),
       layers.Conv2D(64, 3, padding='same', activation='relu'),
       layers.MaxPooling2D(),
       layers.Conv2D(128, 3, padding='same', activation='relu'),
       layers.MaxPooling2D(),
       layers.Dropout(0.7),
       layers.Flatten(),
       layers.Dense(256, activation='relu'),
       layers.Dense(class_num)
     ])
```

compile and train the model

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
[46] model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_7 (Sequential)    (None, 32, 32, 3)         0
_________________________________________________________________
rescaling_5 (Rescaling)      (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 16, 16, 64)        18496
```

ICP5_BDAA (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[51] epochs = 10
     history = model.fit(
       X_train,
       y_train,
       validation_data=(X_test, y_test),
       epochs=epochs,
       batch_size=64,
       )
```

```
Epoch 1/10
782/782 [==============================] - 11s 13ms/step - loss: 1.6901 - accuracy: 0.3793 - val_loss: 1.3816 - val_accuracy: 0.5005
Epoch 2/10
782/782 [==============================] - 10s 13ms/step - loss: 1.4284 - accuracy: 0.4794 - val_loss: 1.2119 - val_accuracy: 0.5581
Epoch 3/10
782/782 [==============================] - 10s 13ms/step - loss: 1.3286 - accuracy: 0.5214 - val_loss: 1.2037 - val_accuracy: 0.5677
Epoch 4/10
782/782 [==============================] - 10s 12ms/step - loss: 1.2602 - accuracy: 0.5491 - val_loss: 1.1659 - val_accuracy: 0.5857
Epoch 5/10
782/782 [==============================] - 10s 13ms/step - loss: 1.2139 - accuracy: 0.5660 - val_loss: 1.0898 - val_accuracy: 0.6138
Epoch 6/10
782/782 [==============================] - 10s 13ms/step - loss: 1.1827 - accuracy: 0.5784 - val_loss: 1.0909 - val_accuracy: 0.6207
Epoch 7/10
782/782 [==============================] - 10s 13ms/step - loss: 1.1510 - accuracy: 0.5909 - val_loss: 1.1035 - val_accuracy: 0.6190
Epoch 8/10
782/782 [==============================] - 10s 13ms/step - loss: 1.1228 - accuracy: 0.6012 - val_loss: 1.0395 - val_accuracy: 0.6389
Epoch 9/10
782/782 [==============================] - 10s 12ms/step - loss: 1.1049 - accuracy: 0.6065 - val_loss: 0.9875 - val_accuracy: 0.6534
Epoch 10/10
782/782 [==============================] - 10s 13ms/step - loss: 1.0857 - accuracy: 0.6122 - val_loss: 0.9452 - val_accuracy: 0.6672
```

```
#model as given has reached = Accuracy: 65.71%
# Accuracy: 51.32% with softmax
# Accuracy: 73.02% with relu activiation layer and conv2d start 32

model_eval = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (model_eval[1]*100))
```

```
Accuracy: 66.72%
```

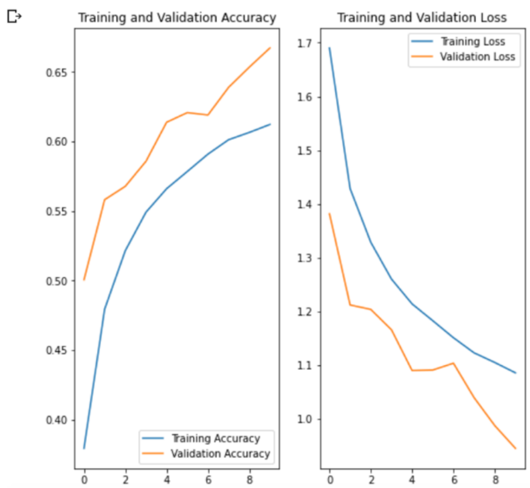
BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

ICP5_BDAA (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Hyper-parameter #4:
(Data Augmentation – preprocessing.RandomFlip("horizontal_and_vertical")
This time I changed the data augmentation in order to help the model to be more aspects of the data and generalize better as well as to help in preventing the overfit. In the randomly flip, I changed the mode to "horizontally and vertically" which mean to flip the images left-tight (horizontal), and top-bottom (vertical). In the end this would not help and the model was still overfitting.

BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

ICP5_BDAA (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

We will implement data augmentation using experimental Keras Preprocessing Layers. These can be included inside your model like other layers, and run on the GPU.

[12]
```
data_augmentation = keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical",
                                                 input_shape=(img_height,
                                                              img_width,
                                                              3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
  ]
)
```

Dropout

Another technique to reduce overfitting is to introduce Dropout to the network, a form of regularization. When you apply Dropout to a layer it randomly drops out (by setting the activation to zero) a number of output units from the layer during the training process. Dropout takes a fractional number as its input value, in the form such as 0.1, 0.2, 0.4, etc. This means dropping out 10%, 20% or 40% of the output units randomly from the applied layer

+ Code + Text

[13]
```
model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(256, activation='relu'),
  layers.Dense(class_num)
])
```

BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

ICP5_BDAA (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[16]
```
epochs = 10
history = model.fit(
  X_train,
  y_train,
  validation_data=(X_test, y_test),
  epochs=epochs,
  batch_size=64,
  )
```

```
Epoch 1/10
782/782 [==============================] - 13s 15ms/step - loss: 1.6988 - accuracy: 0.3732 - val_loss: 1.3912 - val_accuracy: 0.4909
Epoch 2/10
782/782 [==============================] - 11s 15ms/step - loss: 1.4275 - accuracy: 0.4815 - val_loss: 1.3097 - val_accuracy: 0.5317
Epoch 3/10
782/782 [==============================] - 11s 15ms/step - loss: 1.3255 - accuracy: 0.5248 - val_loss: 1.1903 - val_accuracy: 0.5796
Epoch 4/10
782/782 [==============================] - 11s 15ms/step - loss: 1.2443 - accuracy: 0.5545 - val_loss: 1.1928 - val_accuracy: 0.5778
Epoch 5/10
782/782 [==============================] - 11s 14ms/step - loss: 1.1878 - accuracy: 0.5777 - val_loss: 1.0925 - val_accuracy: 0.6058
Epoch 6/10
782/782 [==============================] - 11s 15ms/step - loss: 1.1414 - accuracy: 0.5950 - val_loss: 1.0907 - val_accuracy: 0.6172
Epoch 7/10
782/782 [==============================] - 11s 14ms/step - loss: 1.1050 - accuracy: 0.6068 - val_loss: 1.0149 - val_accuracy: 0.6419
Epoch 8/10
782/782 [==============================] - 11s 15ms/step - loss: 1.0728 - accuracy: 0.6184 - val_loss: 0.9684 - val_accuracy: 0.6548
Epoch 9/10
782/782 [==============================] - 11s 14ms/step - loss: 1.0444 - accuracy: 0.6251 - val_loss: 0.9777 - val_accuracy: 0.6563
Epoch 10/10
782/782 [==============================] - 11s 14ms/step - loss: 1.0271 - accuracy: 0.6352 - val_loss: 0.9421 - val_accuracy: 0.6692
```

```
#model as given has reached = Accuracy: 65.71%
# Accuracy: 51.32% with softmax
# Accuracy: 73.02% with relu activiation layer and conv2d start 32
# Accuracy: 66.92% : with data augmentation mode (horizontal_and_vertical)
model_eval = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (model_eval[1]*100))
```

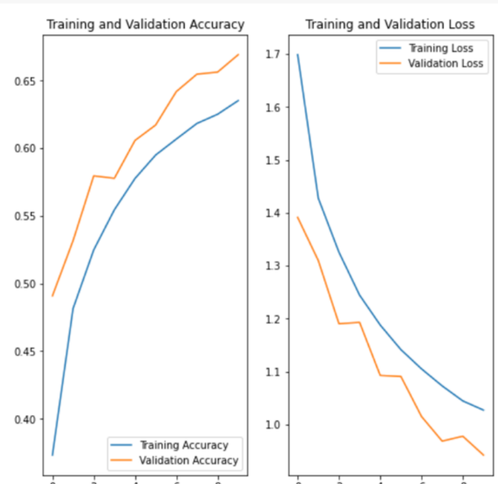
```
Accuracy: 66.92%
```

ICP5_BDAA (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[18]
```
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Predict on new data

BDAA-ICP5-wiki report : Yousef Almutairi
Date: 09/27/2021

5. Video link:

https://drive.google.com/file/d/1bKdPBeNvON77XFZudalbUp7ISlHcyqME/view?usp=sharing