Information Retrieval Systems

Lab Practical and date – Practical 10, Thursday 5th May 2022

Name and Roll Number- Naman Thaker (20BCE529)

Practical Objective- Implement ANN on dataset

Dataset- CIFAR-10

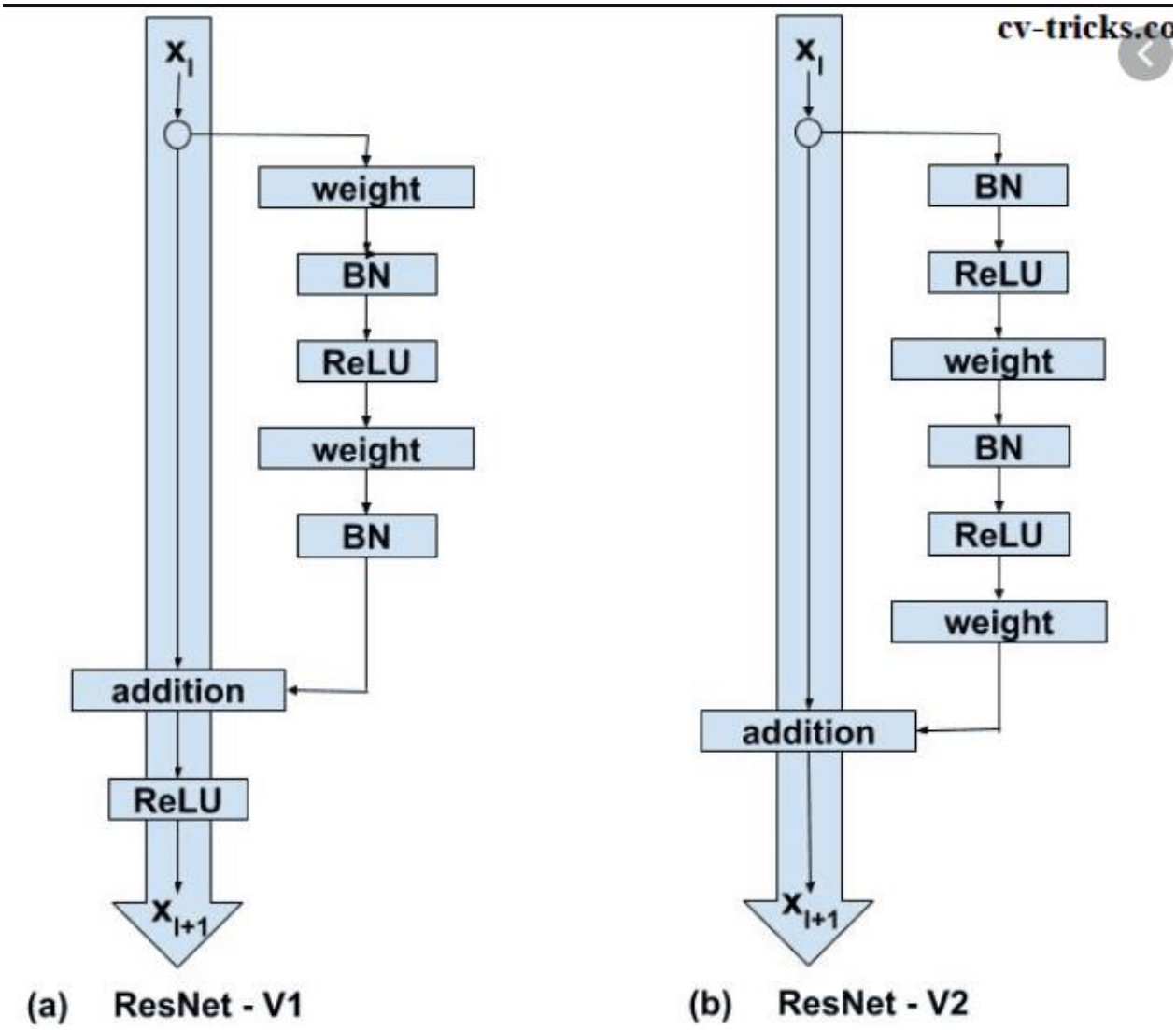
We have used the CIFAR-10 dataset for our training model which is a well-understood dataset and widely used for benchmarking in the field of machine learning.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

10 Classes in the DataSet

ANN Architecture



.For this practical we used the pre trained deep learning model ResNet50V2 and use the standard wrights that is initiated by the imagenet option

Input shape defines the input layer

Include_top specifies whether we want to select the final dense layers or not.

```
base_model = tf.keras.applications.ResNet50V2(input_shape=(32, 32, 3), weights='imagenet', include_top=False)

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(1024),
    tf.keras.layers.ReLU(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

Defining the model

We define each layer, number of neurons, the activation function and the dropout value

Activation function are either Relu or softmax

```
model.compile(
    optimizer='adam',
    loss = 'sparse_categorical_crossentropy',
    metrics=['sparse_categorical_accuracy']
)
model.summary()
```

Compiling the model

We use the adam optimizer to compile the model

Adam optimizer calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

The initial value of the moving averages and beta1 and beta2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

And is the preferred option for most of the deep learning applications.

The loss is the sparse_categorical_crossentropy, which Computes the crossentropy loss between the labels and predictions. We Use this crossentropy loss function since there are two or more label classes. We expect labels to be provided as integers.

Model Summary

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50v2 (Functional)      (None, 1, 1, 2048)        23564800
_________________________________________________________________
global_average_pooling2d_8 ( (None, 2048)              0
_________________________________________________________________
dropout_18 (Dropout)         (None, 2048)              0
_________________________________________________________________
dense_16 (Dense)             (None, 1024)              2098176
_________________________________________________________________
re_lu_8 (ReLU)               (None, 1024)              0
_________________________________________________________________
dropout_19 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                10250
=================================================================
Total params: 25,673,226
Trainable params: 25,627,786
Non-trainable params: 45,440
_________________________________________________________________
```

Model Training- using 10 iterations to optimize the weights in batch sizes of 512

validation_split: Float between 0 and 1. Fraction of the training data to be used as validation data. The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch. The validation data is selected from the last samples in the x and y data provided, before shuffling.

```
model.fit(X_train, y_train, batch_size=512, epochs=10, validation_split=0.1)
```

```
Epoch 1/10
88/88 [==============================] - 10s 116ms/step - loss: 2.0257 - sparse_categorical_accuracy:
Epoch 2/10
88/88 [==============================] - 9s 107ms/step - loss: 1.2969 - sparse_categorical_accuracy: 6
Epoch 3/10
88/88 [==============================] - 9s 107ms/step - loss: 0.9605 - sparse_categorical_accuracy: 6
Epoch 4/10
88/88 [==============================] - 9s 107ms/step - loss: 0.7402 - sparse_categorical_accuracy: 6
Epoch 5/10
88/88 [==============================] - 9s 107ms/step - loss: 0.6127 - sparse_categorical_accuracy: 6
Epoch 6/10
88/88 [==============================] - 9s 107ms/step - loss: 0.5174 - sparse_categorical_accuracy: 6
```

Calculate the model accuracy

```
model.evaluate(X_test, y_test, batch_size=512)
```

```
20/20 [==============================] - 1s 31ms/step - loss: 0.9140 - sparse_categorical_accuracy: 0.7358
[0.9140002727508545, 0.73580002784729]
```

We got an accuracy of 73.5 percent

Sample Input

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images..

Simple Output

```
model.evaluate(X_test, y_test, batch_size=512)
```

```
20/20 [==============================] - 1s 31ms/step - loss: 0.9140 - sparse_categorical_accuracy: 0.7358
[0.9140002727508545, 0.73580002784729]
```

. Accuracy of 73.5%.

Conclusion:

In this practical, We have successfully explored the extensions of a baseline model to improve its learning and model capacity.

The model converges well for about 75 or 80 epochs, at which point there is no further improvement on the test dataset. with a decent accuracy of 73% it would be better if We could elaborate upon this model and add early stopping with a patience of about 10 epochs to save a well-performing model on the test set during training at around the point that no further improvements are observed.