Project-7

CIS 666 Artificial Intelligence

Convolutional Neural Network (CNN) Implementation on COVID-19 dataset

Fathima Syeda

Student ID: 2790024

Abstract

In this project we investigate the detecting of COVID-19 in CT scans (images) with Keras, TensorFlow, and Deep Learning. We make use of a simple Deep Convolution network to build a model with 2 alternate convolution and pooling layers followed by fully connected layer. To improve the performance, we make use of the He_normal initializer for weight initialization coupled with the ReLU Activation function. The model is built with various optimizers – Adam, SGD (Stochastic Gradient Descent), RMSProp Root mean square) and the performances are compared.

Contents

1. Introduction:

A convolutional neural network is a special kind of Artificial neural network that specialises in detecting patterns. A CNN network consists of three kinds of layers: Convolutional layer, Pooling Layer, Fully connected Layer.

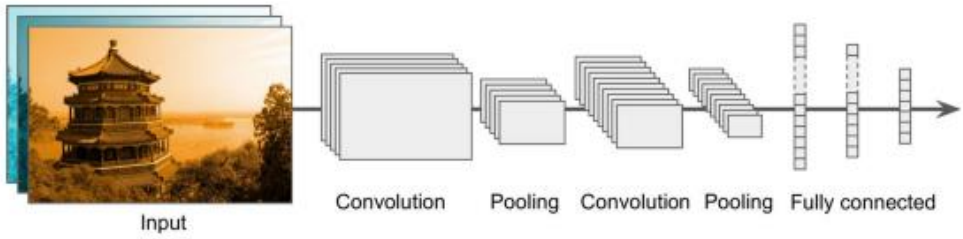


Figure 14-11. Typical CNN architecture

Convolutional Layer:

The most important building block of a CNN is the convolutional layer. neurons in the first convolutional layer are not connected to every single pixel in the input image but only to pixels in their receptive fields. In turn, each neuron in the second convolutional layer is connected only to neurons located within a small rectangle in the first layer. This architecture allows the network to concentrate on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next hidden layer, and so on.

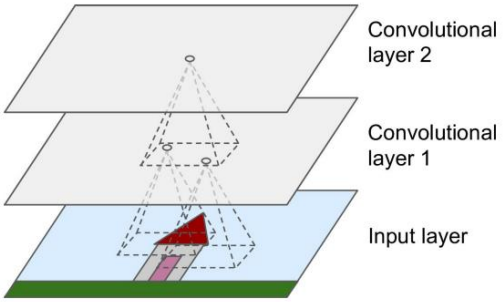


Figure 14-2. CNN layers with rectangular local receptive fields

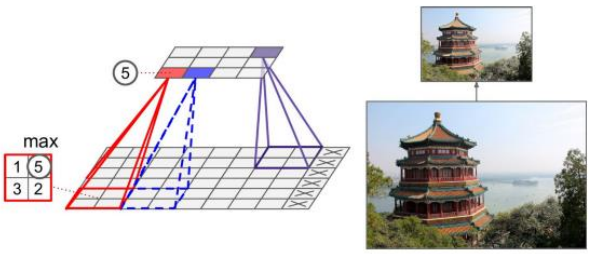


Figure 14-8. Max pooling layer (2 × 2 pooling kernel, stride 2, no padding)

Pooling Layers:

The goal of these layers is to subsample (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters (thereby limiting the risk of overfitting). Just like in convolutional layers, each neuron in a pooling layer is connected to the outputs of a limited number of neurons in the previous layer, located within a small rectangular receptive field.

Fully Connected Layer:

This layer is heavy data driven layer . It gives the top best or most probable classification. In this layer the actual learning of the non-linear combination of the features occurs.

Batch Normalization

This operation can significantly reduce the danger of the vanishing/ exploding gradients problems. It lets the model learn the optimal scale and mean of each of the layer's inputs. This operation simply zero-centres and normalizes each input, then scales and shifts the result using two new parameter vectors per layer: one for scaling, the other for shifting.

Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (pp.339, 456-457). O'Reilly Media. Kindle Edition

2. Steps involved in Convolutional Neural Networks.

1. Read the positive and negative samples from the Covid -19 dataset
2. Set aside 70 % of the data from each of the 15 categories of the scene dataset for training and the rest are used for testing.
3. Resize the images of the scene dataset to 32x32x1.
4. Set up the parameters for the Convolution layer 1 as follows:
    a. Filters dimension=[3,3,D], Input=[wxHxD], **number of filters =32**, stride =1
    b. Set the activation function as inside the convolution layer as **ReLu**.
5. Set up the parameters for the Pooling Layer as **Max Pooling : 2x2 with stride =2**
6. Again, add another Convolution layer with the same parameters as the first one but the **number of filters =64** and the input is the result of pooling from the previous layer.
7. Add another Pooling layer with the same parameters as those in the previous one.
8. After each pooling layer add a batch_normalization() layer to deal with vanishing gradient descent problem and a dropout layer of 25% to reduce overfitting
9. Finally add three Dense() of 1024,512,2 to form the Fully Connected Layer.
10. Compile the network to train over the CNN and test the data on the model to get the accuracy.

3. Program Outline

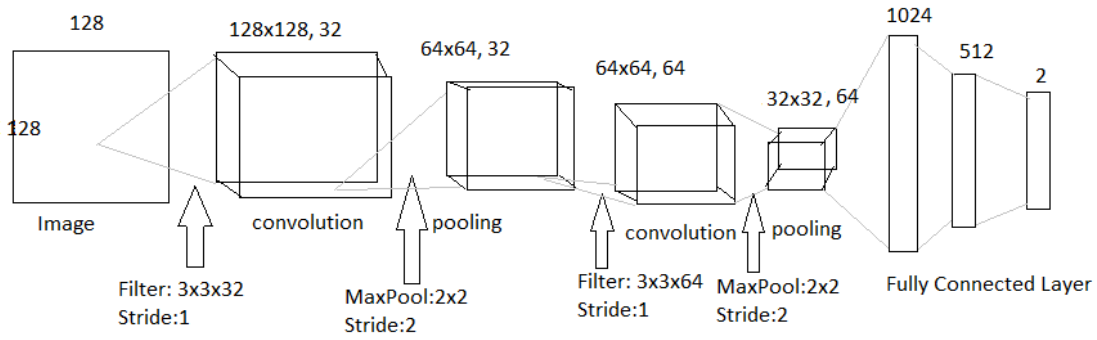
In this project we make use of python's TensorFlow and Keras Deep learning library to make the CNN network to investigate the covid-19 dataset.

The following is the structure of the CNN network built.

The CNN is built with 2 convolution layers and 2 MaxPooling layers along with the Fully Connected layer. We also add 2 Batch Normalization layers to deal with vanishing gradient descent problem and Dropout layers to stop overfitting of model after every hidden layer.

Layer	Type	Size	Num of Kernels	Kernel size	Stride	Activation
IN	Input	128x128	-	-	-	-
C1	Convolution	128x128	32	3x3	1	ReLU
S2	MaxPooling	64x64	32	2x2	2	-
B3	batch_normalization	64x64	32			

D4	Dropout-0.25	64x64	32			
C5	Convolution	64x64	64	3x3	1	ReLU
S6	MaxPooling	32x32	64	2x2	2	-
B7	batch_normalization	32x32	64			
D8	Dropout-0.25	32x32	64			
F9	Fully Connected	1024	-	-	-	ReLU
D10	Dropout-0.25	1024				
F11	Fully Connected	512	-	-	-	ReLU
D12	Dropout-0.25	512	-	-	-	
Out	Fully Connected	2	-	-	-	SoftMax



The above figure shows the structure of a simple CNN for the Covid 19 dataset.

The CNN model consists of a convolution layer with parameters as filters = 32, kernel size=(3,3), padding = 'Same', input shape=28x28x1 stride=1 . An activation layer of ReLU is included in the convolution layer itself. This layer outputs a 26x26x32 convoluted image. The next layer is a polling layer , where we use a MaxPool :2x2 with stride =2 whose output is 13x13x32. The third layer is again a convolution layer where the number of filters is 64 . Then again, a second MaxPool layer is added followed by a Fully connected layer made of Dense(1024), Dense(512), Dense(2) with activation function as SoftMax. We initialize the weights by making use of the He_normal initializer and use the ReLU activation function in the fully connected layer to improve the performance of the model.

4 Obtained Results

The following is the result when the CNN network is used to train and model **the COVID-19 Dataset** with 2 alternate convolution and pooling layers with filters =32,64 with epoch =10.

---

1. Optimizer used=Adam

Epoch 1/12
885/885 [==============================] - 40s 45ms/step - loss: 1.0785 - acc: 0.9220
Epoch 2/12
885/885 [==============================] - 39s 44ms/step - loss: 0.9927 - acc: 0.9333
Epoch 3/12
885/885 [==============================] - 39s 44ms/step - loss: 0.4866 - acc: 0.9684
Epoch 4/12
885/885 [==============================] - 39s 44ms/step - loss: 0.5561 - acc: 0.9638
Epoch 5/12
885/885 [==============================] - 39s 44ms/step - loss: 0.5992 - acc: 0.9605
Epoch 6/12
885/885 [==============================] - 38s 43ms/step - loss: 1.0253 - acc: 0.9333
Epoch 7/12
885/885 [==============================] - 39s 44ms/step - loss: 0.8944 - acc: 0.9412
Epoch 8/12
885/885 [==============================] - 39s 44ms/step - loss: 1.0434 - acc: 0.9322
Epoch 9/12
885/885 [==============================] - 38s 43ms/step - loss: 0.6952 - acc: 0.9548
Epoch 10/12
885/885 [==============================] - 38s 43ms/step - loss: 0.7461 - acc: 0.9514
Epoch 11/12
885/885 [==============================] - 39s 44ms/step - loss: 0.7820 - acc: 0.9492
Epoch 12/12
885/885 [==============================] - 39s 44ms/step - loss: 0.9558 - acc: 0.9379
379/379 [==============================] - 2s 7ms/step

Accuracy: 0.8601583242416382

---

2. Optimizer used=SGD

Epoch 1/12
885/885 [==============================] - 28s 31ms/step - loss: 0.5109 - acc: 0.9379
Epoch 2/12
885/885 [==============================] - 27s 31ms/step - loss: 0.1466 - acc: 0.9831
Epoch 3/12
885/885 [==============================] - 27s 31ms/step - loss: 0.0934 - acc: 0.9921
Epoch 4/12
885/885 [==============================] - 27s 30ms/step - loss: 0.0819 - acc: 0.9910
Epoch 5/12
885/885 [==============================] - 27s 30ms/step - loss: 0.0341 - acc: 0.9944
Epoch 6/12
885/885 [==============================] - 27s 30ms/step - loss: 0.0149 - acc: 0.9944
Epoch 7/12
885/885 [==============================] - 27s 30ms/step - loss: 9.3274e-04 - acc: 0.9989
Epoch 8/12
885/885 [==============================] - 27s 31ms/step - loss: 4.4748e-04 - acc: 1.0000
Epoch 9/12
885/885 [==============================] - 27s 31ms/step - loss: 0.0333 - acc: 0.9932
Epoch 10/12
885/885 [==============================] - 27s 31ms/step - loss: 4.3607e-04 - acc: 1.0000

Epoch 11/12
885/885 [==============================] - 27s 30ms/step - loss: 8.6494e-04 - acc: 1.0000
Epoch 12/12
885/885 [==============================] - 27s 31ms/step - loss: 0.0019 - acc: 0.9989
379/379 [==============================] - 3s 7ms/step

Accuracy: 0.9894459247589111

---

3. **Optimizer used=RMSProp**

Epoch 1/12
885/885 [==============================] - 34s 39ms/step - loss: 3.3050 - acc: 0.7672
Epoch 2/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 3/12
885/885 [==============================] - 33s 37ms/step - loss: 3.2670 - acc: 0.7876
Epoch 4/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 5/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 6/12
885/885 [==============================] - 35s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 7/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 8/12
885/885 [==============================] - 35s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 9/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 10/12
885/885 [==============================] - 35s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 11/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
Epoch 12/12
885/885 [==============================] - 34s 39ms/step - loss: 3.2670 - acc: 0.7876
379/379 [==============================] - 3s 7ms/step

Testing Accuracy: 0.7994722723960876

TASK 3:

The following is the structure of the model

Model: "sequential_18"
_____
Layer (type) Output Shape Param #
===
conv2d_32 (Conv2D) (None, 128, 128, 32) 320
_____
max_pooling2d_31 (MaxPooling (None, 64, 64, 32) 0
_____
batch_normalization_23 (Batc (None, 64, 64, 32) 128
_____
dropout_41 (Dropout) (None, 64, 64, 32) 0
_____
conv2d_33 (Conv2D) (None, 64, 64, 64) 18496
_____

```
max_pooling2d_32 (MaxPooling (None, 32, 32, 64)       0
_________________________________________________________________
batch_normalization_24 (Batc (None, 32, 32, 64)       256
_________________________________________________________________
dropout_42 (Dropout)         (None, 32, 32, 64)       0
_________________________________________________________________
flatten_19 (Flatten)         (None, 65536)            0
_________________________________________________________________
dense_45 (Dense)             (None, 1024)             67109888
_________________________________________________________________
dropout_43 (Dropout)         (None, 1024)             0
_________________________________________________________________
dense_46 (Dense)             (None, 512)              524800
_________________________________________________________________
dropout_44 (Dropout)         (None, 512)              0
_________________________________________________________________
dense_47 (Dense)             (None, 2)                1026
=================================================================
Total params: 67,654,914
Trainable params: 67,654,722
Non-trainable params: 192
_________________________________________________________________
```

5 Conclusion

- The optimizer SGD yields better results than Adam which yields better results than the RMSProp optimiser.
- Using the He initializer in pair with the ReLU activation for weight initialization gives better accuracy than when the default initializer glorot is used.

6. References

- Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (pp. 456-457). O'Reilly Media. Kindle Edition
- Class Notes