Lab report – TNM112

Deep learning for media technology, Lab 2

Johan Bäcklund (johba008)

Lab partner: Gabriel Cederqvist (gabce093), Johan Hedin (johed658)

Wednesday 13th December, 2023 (10:18)

Abstract

This lab involves implementing and experimenting with settings for a CNN network. The implementation will be compared with a model that uses the Keras library. An attempt to find suitable settings for a CNN will be performed. One data sets consists of images that contain written numbers. Another data set consists of medical images that may contain lymph nodes. The results of this experiment appears to indicate that augmentations of images is a powerful tool when a limited number of images exist in the data set.

1 Introduction

A convolutional neural network (CNN) is feed-forward neural network where the architecture consists of layers that utilize kernel operations in convolution layers. This has applications for image classification purposes. The convolution layers can be considered as a way to extract features of an image. These features can then finally be processed by a dense layer at the end of the model. In this case the dense layer is an MLP.

In this lab, a set of tasks will be performed that involves implementing a CNN, as well as experimenting with different features and settings that affect the accuracy of a model.

2 Background

The convolutional layer in the network can be described by equation 1.

$$H_j^{(l)} = \sigma(\sum_{i=1}^{C_I} W_{i,j}^{(l)} * H_i^{(l-1)} + b_j^{(l)}) \tag{1}$$

Where H is a $[D_y\ D_x\ C_O\ ]$ tensor where D_x and D_y corresponds to the width and height of an image respectively. C_O corresponds to the channels in the layer. W is a $[R\ S\ C_I\ C_O\ ]$ tensor where $C_I\ C_O$ is the filter kernels that has the height R and width S.

This function assumes that the images in different color channels have the same resolution. If this isn't the case, images may be padded to consistently work with channels that vary in resolution.

3 Method

The following tasks are taken from lab instructions in [1].

3.1 Task 1

The first task involves implementing a CNN, and compare the implementation to a trained model that was built using the Keras library. Components that will be implemented consists of a *Convolutional*, *Max Pooling*, *Flatten* and *Dense* layer. The *Dense* layer, as well as activation functions, are reused from Lab 1.

The dataset consists of binary images that represent written numbers. The accuracy and cross-entropy loss was compared to the trained model. The implementation is deemed to be valid if it has similar results.

3.2 Task 2

The second task involves using our own implementation of a CNN from 3.1 but only training it with a subset of the previous dataset, namely, 128 images. Creating an accurate model with the small dataset will become a challenge. To fully utilize the dataset, we can apply regularization strategies to mitigate this problem.

The images will be augmented during training. In essence the augmented images can be considered to be new images, which addresses the problem of smaller a dataset. The augmentation operations that are experimented with in this instance are rotation, translation and zoom. Suitable augmentations when training the model depends on the properties of the images in the dataset. Since we know that the images corresponds to written numbers, it would be unwise to flip the images or rotate them more than 90 degrees. This would cause the resulting images to not properly represent a written number. It is also important to not translate or zoom the images with a high factor. This could cause part of the numbers to be translated outside the bounds of the original image.

To prevent overfitting, a dropout operation may be used in the dense layer. The idea is that a specific neuron in the network may gain a higher weight compared to other neurons during training, which would overrule the predictions of other neurons. By randomly "dropping" a set of neurons for each step when training, it is probable that such a neuron will get dropped during training.

Another technique that addresses this problem is batch normalization. By applying batch normalization to a layer, the outputs of the layer will be transformed in attempt to maintain the mean output to be *0*, and the standard deviation to be *1*. A neuron with a high weight may then be compressed, while amplifying the impact of low weight neurons.

3.3 Task 3

The third task is similar to 3.2, and involves setting up a model that classifies lymph nodes in medical images. This task is different compared to 3.2, since the images do not consist of binary values, but also uses 3 color channels in the *RGB* color space. This allows the possibility for contrast and brightness augmentations, but also increased the time it takes to train the model.

The settings of the model was selected through experimentation in the majority of cases, although, the contents of the images were taken into consideration. In this case, rotating or flipping the image in any direction would still represent a valid image in the dataset, which allows for more variation in augmentation.

The model will output a true or false statement, where it classifies whether the image contains a lymph node or not. The model is deemed valid if it has an accuracy above 50%. This would imply that the model is better at classifying a lymph node compared to making a random guess.

The layers that were chosen for the model is shown in the following code snippet:

```
x = layers.Input(shape=data.x_train.shape[1:])

aug_flip = layers.RandomFlip(mode="horizontal_and_vertical")(x)

conv1 = conv_block(aug_flip, N=2, channels=8, kernel_size=(3, 3),
    activation='relu', padding='same')

pool1 = layers.MaxPooling2D()(conv1)

conv2 = conv_block(pool1, N=2, channels=16, kernel_size=(3, 3),
    activation='relu', padding='same')

flat1 = layers.Flatten()(conv2)

dense1 = layers.Dense(512, kernel_regularizer=regularizers.L1L2
    (0.0002, 0.001),  activation='relu')(flat1)

dense2 = layers.Dense(512, kernel_regularizer=regularizers.L1L2
    (0.0002, 0.001), activation='relu')(dense1)

drop2 = layers.Dropout(0.02)(dense2)

dense3 = layers.Dense(512, kernel_regularizer=regularizers.L1L2
    (0.0002, 0.001),  activation='relu')(drop2)

drop3 = layers.Dropout(0.02)(dense3)

dense4 = layers.Dense(512, kernel_regularizer=regularizers.L1L2
    (0.0002, 0.001),  activation='relu')(drop3)

y = layers.Dense(data.K, activation='softmax')(dense4)
```

4 Results

4.1 Task 1

The absolute difference between the trained model and our implementation was around *0.00008314*. This value can be considered close to zero, and entails that the implementation is valid since the models are similar.

4.2 Task 2

The model was trained 5 times, and gave the following results in terms of accuracy:

Mean Accuracy
0.88669997
0.91869998
0.93409997
0.94260001
0.90570003

This resulted in a mean accuracy about *0.91755999* and mean variance about *0.00039897*.

4.3 Task 3

By selecting the settings of the model experimentally, the model was able to achieve an accuracy around *79%*.

5 Conclusion

Augmentations of the images provided the most significant increase in accuracy for all models. This was especially the case for task 2 in 3.2. This could be explained by the fact that the data set only consisted of 128 images.

Task 3 in 3.3 proved to be difficult when finding a good set of parameters and settings. Some specific augmentation operations appeared to be ineffective and sometimes counterproductive, such as rotating the images. An explanation for this could be that the images are small, which results in information being lost when performing a rotation operation. The rotation would result in pixels of the images being an interpolation of a group of pixels. Since the images are small, this causes a coarse transformation in the images.

For task 3, augmenting the brightness and contrast did not appear to have as significant of effect as expected. An observation of a set of images in the dataset were made. Images that contained a lymph node often appeared more purple. This doesn't necessarily imply that purple areas are caused by lymph nodes, but perhaps that lymph nodes appear in purple areas. If the images were augmented in a way that allowed the model to distinguish purple areas it is possible that the accuracy could increase. An augmentation that could be further explored would be to change the tonal range of the image to increase the contrast specifically in the range between the purple color and white color.

Use of generative AI

Generative AI was not used during the project or report

References

[1] TNM112. Lab 2 – convolutional neural networks, 2023. Lab description – Deep learning for media technology.