236606 - Deep Learning - Homework 3

Eyal Ben David - 305057416

June 5, 2018

Convolutional Neural Network

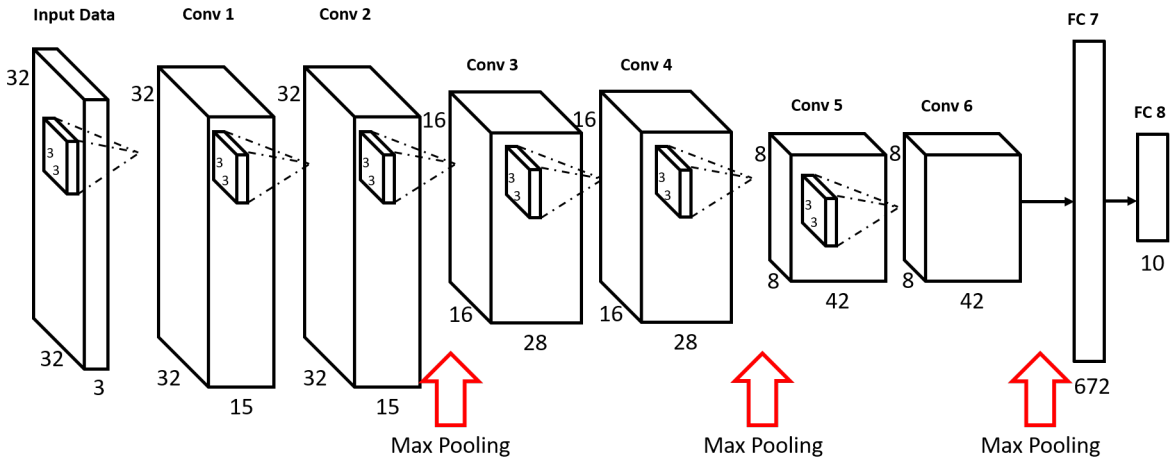
During our assignment we were asked to build a cifar10 classifier with a maximum of 50K parameters, and a min accuracy of - 85%. We tried multiple types off architecture's in order to meet with these requirements including; very deep (9-11 layers) convolutional networks with shallow chanels (8-16), shalow convlolutional layers (3-4 layers) with dwide channels(48-64) procceeded by wide dense layers and . We also explored different optimizers such as adam, rmsprop and sgd and different regularization technique like L1, L2, dropout and data ogmentation. Our winning architecrture is a fully convolutional architecture with six convolutional layers, with 'ReLu' activation and increasing channel width succeeded by a softmax layer. Here is our model summary:

Figure 1: Model summary of our chosen model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 15)        420
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 15)        60
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 15)        2040
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 15)        60
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 15)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 28)        3808
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 28)        112
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 28)        7084
_________________________________________________________________
```

```
batch_normalization_4 (Batch (None, 16, 16, 28)        112
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 28)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 42)          10626
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 42)          168
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 42)          15918
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 42)          168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 42)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 672)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 672)               0
_________________________________________________________________
batch_normalization_7 (Batch (None, 672)               2688
_________________________________________________________________
dense_1 (Dense)              (None, 10)                6730
=================================================================
Total params: 49,994
Trainable params: 48,310
Non-trainable params: 1,684
_________________________________________________________________
```

Figure 2: Model visualization of our chosen model



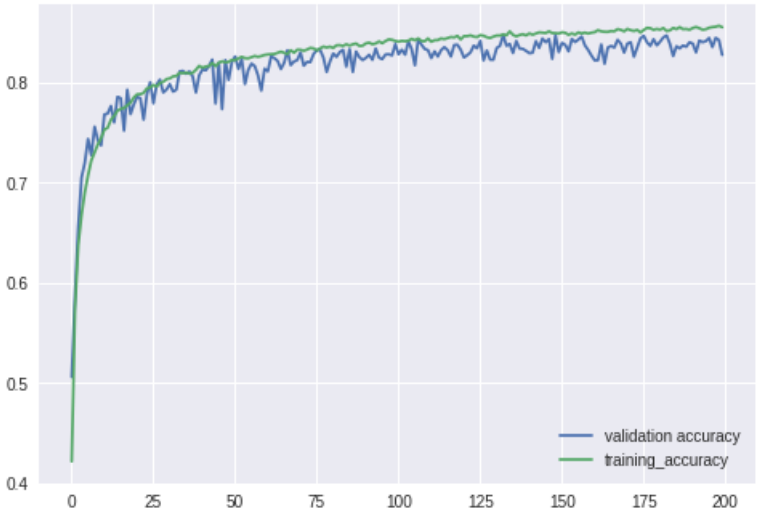
To train this model we used the 'adam' optimizer with Keras default parameters

for 400 epochs. Another tecnichue from keras which gave us a good boost for the accuracy was the 'Data Generator', which we used in order to augment the data and decrease overfiting of our CNN. The final results of this model are presented in 'table 1' and the training graph is ploted in 'figure 2'. The trained model is submited together with the code in the .zip file.

Table 1: Chosen model results for 'Cifar10' dataset

	Loss	**Accuracy**
Train	**0.30948**	**0.8922**
Test	**0.44967**	**0.8536**

Figure 2: Training accuracy graph of the model in each epoch



Transfer Learning

In this question we have investigated some transfer learning aproaches that include: 'Fine Tuning', 'KNN', 'SVM' (our flavor). The original model is a VGG16 model trained on 'Cifar100' dataset. For adapting to the 'Cifar10' dataset we poped the last layer of the original model, and for each section we added a new layer. The number of examples was hardcoded inside the file. For each aproach we have trained the model with 100, 1000 and 10000 examples.

Transfer learning by Fine Tuning

We took a CIFAR-100 trained model and replaced the last fully-conected layer with a new fully-conected layer that is adapted to ten classes output vector. We then trained only the last layer using 100, 1000, and 10000 examples on CIFAR-10 dataset, while freezing other layers of the model.

Transfer learning with KNN

In this aproach we map the new CIFAR-10 dataset to the domain that was trained by the CIFAR-100 dataset. After passing the data in the pre-trained model we get vectors in dimension of the last output layer - 512, then we aply KNN to the 5 closest neighbours. This can work because te classes of CIFAR-10 and CIFAR-100 have a strong correlation between them. We used 5 closest neighbours because it gave best result's of all other 'K' options we have tried.

Transfer learning with SVM

In this section we needed to choose our own solution, we chose to use SVM. Similarly to the KNN method, we use the pretrained model of CIFAR-100 dataset, without it's last layer. We pass the new CIFAR-10 data through the model and get a new representation for each example. Then we train these new vectors with a non linear classifier (we are using gaussian kernel) to clasiffy 10 different classes. In each stage we trained the non-linear classifier with different amount of examples. The motivation for this aprroach is that if the previous layers did a good job embeding the data to a new dimension which the data is seperated in this dimension, then it might be classified easily by the non-linear classifier. SVM can also use other kernels in order to use a non linear classifier for the data after it is projected on the new dimension.

Results - Transfer learning

We summarize the result's of this question in 'table 2'.

Table 2: Accuracy for each Transfer Learning model trained on different number of examples

	Fine Tune Train	Fine Tune Test	KNN (neighbours = 5)	SVM
100 examples	70.3%	49.25%	43.15%	48.35%
1,000 examples	56.5%	56.2%	50.88%	57.01%
10,000 examples	61.2%	59.1%	57.51%	62.41%

Coclusions

From these results we can learn couple of things:The more examples we use to train, the accuracy is higher and overfiting decreases. Also, the 'SVM' and 'Fine Tuning' methods work better for transfer learning with this model than 'KNN' method. For each method we trained only the last layer of the model, while we freezed other layers in the model that were trained to classify the CIFAR-100 dataset. Hence it is clear that the more correlation we get between the two dataset, their types of classes, the classes distribution and the similarity of the data itself, the better the results can get. We see that the results are pretty good

in the case of using a pretrained CIFAR-100 model on a CIFAR-10 dataset, and that clariffies that the two dataset are correlated.