Indoor scene recognition using convolutional neural network

Introduction:

Indoor scene recognition is a challenging problem, because indoor scenes are not usually distinguished by their spatial properties, but by the objects they contain. For instance, a library and a warehouse have very similar spatial disposition (shelves with objects), but while a library should contain only books, a warehouse can contain any kind of objects[1].

As a dataset we will use a collection of more than 15620 images divided into 67 classes, which is provided by MIT and available here. The number of images per class varies from 101 to 734 samples. In addition, the width and height of the images is not uniform.

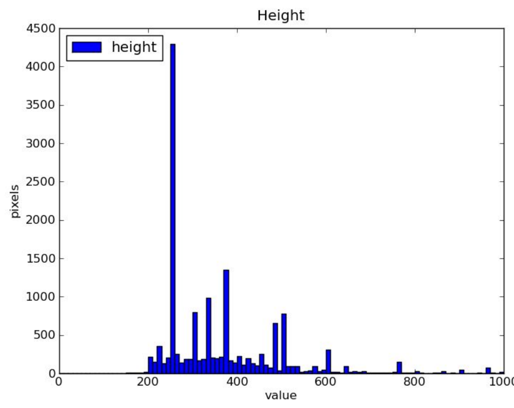
Preprocessing:

Reshaping:

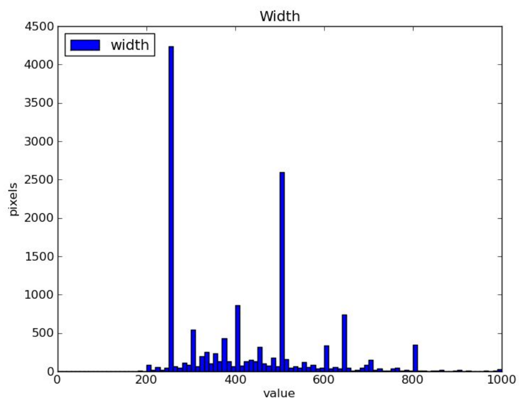
Since our neural network will need a consistent shape of the images, we are forced reshape them before passing them to the network. Two main methods we can use to decrease the width and height of the images are cropping and squashing. The cropping means to cut a part (usually centre) of the original image to achieve the desired shape. The squashing is compressing the original width and height to the desired shape. The disadvantage of cropping is losing some complete part of the information (usually on the edges of the image). The disadvantage of the squashing is the addition of misleading information about the shapes of the images.

Width and height histogram:

In order to find suitable values of width and height to use as input of our CNN we will plot a histogram of widths and heights of our images.



Histogram for height with 50 buckets (x=pixels, y=number of pictures)



Histogram for width with 50 buckets (x=pixels, y=number of pictures)

We can clearly see, that most of the pictures have a resolution between 200 and 400 pictures. The modus of the dataset is 256 pixels for both height and width (the peak of both graphs is at 256).

The significant variation of width and height among the pictures is a reason, why we decided to use squashing to 256x256, instead of cropping. By cutting more than half of the image we might cut out significant objects from the image, thus making the correct classification impossible.

Splitting the dataset into train, validation and test:

The classes in our dataset have different number of samples. We will have to tackle this problem, because the network might be tempted to classify any image as the most common class.

First we will find out how many images we have in our disposal for each class. We will execute following bash script inside our folder with all classes: *find ./ -xdev -type d -print0 | while IFS= read -d '' dir; do echo "$(find "$dir" -maxdepth 1 -print0 | grep -zc .) $dir" done | sort -rn. # Script includes . as a file*

In order to tackle the different number of samples per class we will use an approach called undersampling. We will randomly eliminate the number of images in classes, that have more images than the minimum number of images. We will use an exact random seed, so we can reproduce our splitting in the future, if it is necessary. We could also use oversampling, but this approach might affect the accuracy on different classes, because of different number of real samples.

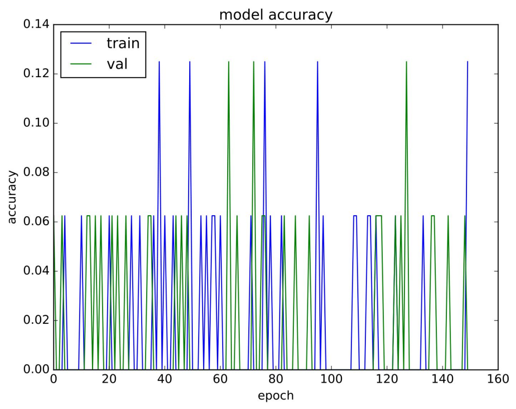
We will split our data into three groups - 75% for training, 15% for validation and 10% for testing. We will use keras structure convention, which will enable keras to deduce class names from directory names.

Initial experiments:

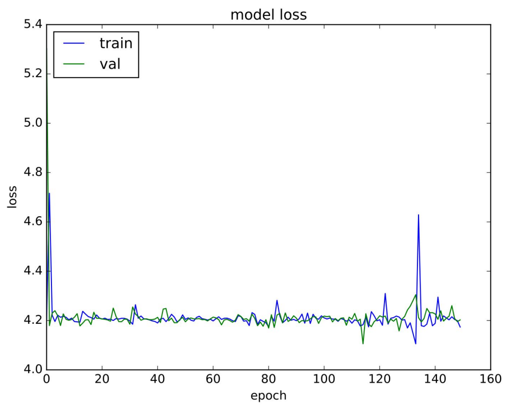
At the beginning of our experiments we will use a simple CNN architecture with small batch sizes, so we can see if our network is actually learning something or not. We will try to classify 67 classes, because of uneven image samplings we will use undersampling to get a sample of 101 images per class. For each class we will get 76 training images, 15 validation images and 10 test images.

As initial network architecture I used the layer composition from guided lab experiments:

1. Two CNN layers with 32 and 64 neurons. Each followed by 2D max pooling.
2. Output of CNN layers is flattened and forwarded to dense 128 neuron layer.
3. All these layers are using relu activation function.
4. At the end of network we have softmax classifier.
5. Our optimization method is stochastic gradient descent and loss is calculated by categorical cross entropy.
6. We will start training with 64 batch size, 150 epochs, 5092 training sample size and 1005 validation sample size.
7. All unmentioned parameters are set to keras default values.



Model accuracy on train and validation dataset



Model loss on train and validation dataset

We can see that network is unable to capture important features of the images. This can contributed to the small number of images per class and large number of classes (67 classes each with 76 pictures).

After several more unsuccessful experiments we decided to decrease the number of classes. We chose the classes with most images, so the network has as much data as possible.

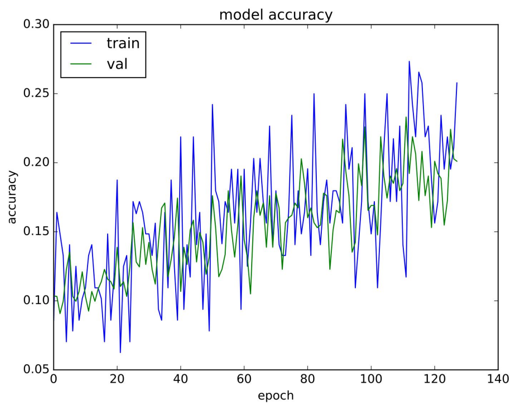
First successful experiments with 10 classes:

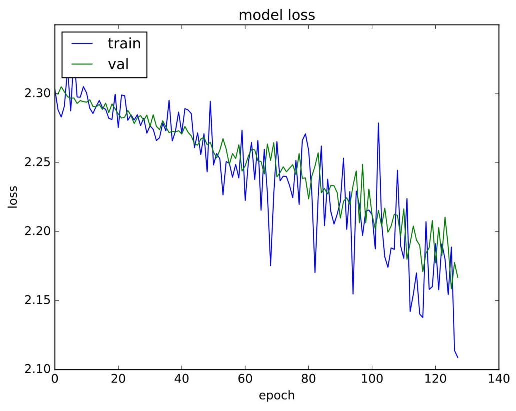
Eventually we decided to use 10 classes, because there is considerable difference in number of images between 10th and 11th class (drop from 457 to 401 images). Alternatively we could have also 15 classes (drop from 346 to 276 images between 15th and 16th class).

For our experiments we have therefore chosen following 10 classes: airport_inside, bar, bedroom, casino, inside_subway, kitchen, livingroom, restaurant, subway and warehouse. After splitting the datasets we get 343 train images, 70 validation images and 47 test images

Network architecture remained the same, but experiment parameters were adjusted.

1. We are using batch size 32, 128 epochs 2048 training sample size and 512 validation sample size.





Model accuracy on train and validation dataset

Model loss on train and validation dataset

We can clearly see a drop in loss and improvement in the accuracy of our model. In addition, we can notice the spikes in the graph, which are probably caused by high learning rate or small batch size.

As we can clearly see an improvement in accuracy of the model we decided to find out the accuracy of the model for each class on the test dataset.

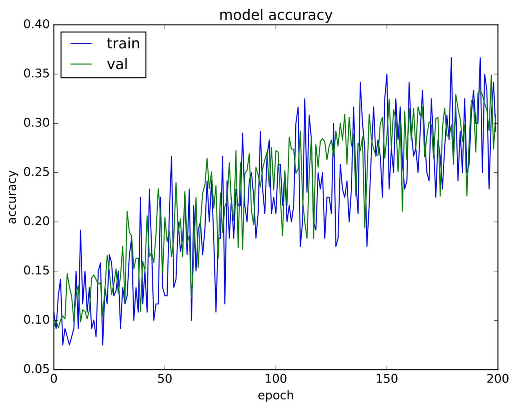
Accuracy per class on test images:

Class airport_inside accuracy: 0.08695652173913043
Class bar accuracy: 0.06521739130434782
Class bedroom accuracy: 0.41304347826086957
Class casino accuracy: 0.3695652173913043
Class inside_subway accuracy: 0.6521739130434783
Class kitchen accuracy: 0.0
Class livingroom accuracy: 0.2391304347826087
Class restaurant accuracy: 0.043478260869565216
Class subway accuracy: 0.021739130434782608
Class warehouse accuracy: 0.13043478260869565
Acc: 0.2017353579175705

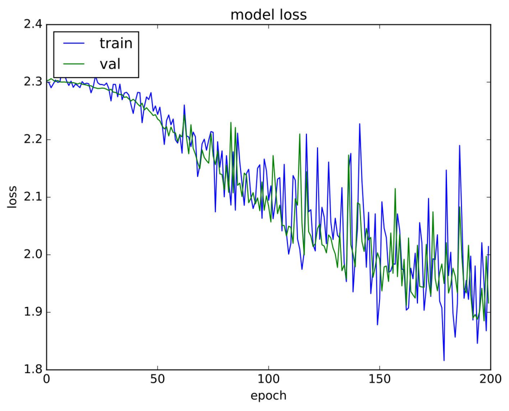
We can clearly see unbalanced distribution of the accuracy among all classes, this was probably caused by a tensorflow warning - epoch compromised. After several experiments we have adjusted the network parameters and balanced the accuracies.

Balancing class accuracy:

We have managed to achieve more balanced accuracy for all classes by adjusting the batch size to 40, number of train samples to 3360 and number of validation samples to 640 samples. The network architecture remained the same as in previous architectures.



Model accuracy on train and validation dataset



Model loss on train and validation dataset

Accuracy per class on test images:

```
Class  airport_inside accuracy:  0.34782608695652173
Class  bar accuracy:  0.5
Class  bedroom accuracy:  0.15217391304347827
Class  casino accuracy:  0.41304347826086957
Class  inside_subway accuracy:  0.34782608695652173
Class  kitchen accuracy:  0.17391304347826086
Class  livingroom accuracy:  0.30434782608695654
Class  restaurant accuracy:  0.32608695652173914
Class  subway accuracy:  0.15217391304347827
Class  warehouse accuracy:  0.391304347826087
Acc:  0.3123644251626898
```

Optimization of the network parameters, which was done in order to balance the accuracy on the different classes, has also increased the accuracy by about 5% on the epoch 128 in comparison with previous experiment.

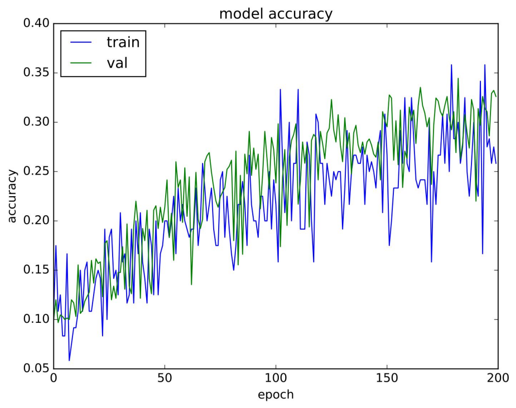
This was achieved by the equal number of images for all classes in each epoch. The parameters were chosen, so they are divisible by the number of classes (10) and by the size of the batch size (40). The size of batch was chosen to be close to previous batch size, so we could compare new experiments with previous experiments.

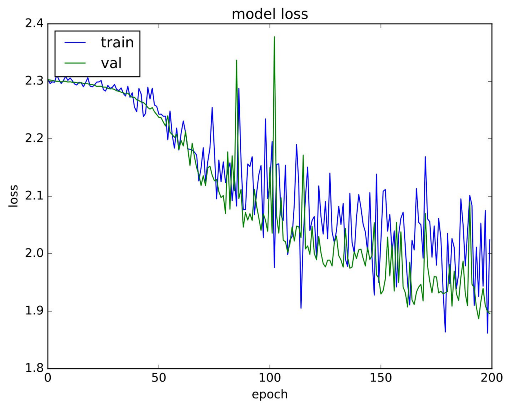
In order to increase the accuracy of the model we decided to try several methods - increase the amount of training data using **image augmentation,** increase the number of **epochs** as we have not reached overfitting yet (it has a drawback of increase in the time of every experiment), **change of the activation function** from rectified linear unit to leaky rectified linear units or exponential linear units and increasing the **number of layers and number of neurons** in them.

Image augmentation using Keras image data generator:

Since we posses by only 343 training images per class, which is not enough data for deep learning algorithm, we can generate new images from the images we have. This process is called data augmentation.

We have multiple image transformation options in image augmentation. We have decided not to use any colour adjusting transformation as indoor scenes might very well be defined by their colours. We used rotations of the pictures in the range of +15 to -15 degrees, width and height shifts of up to 10% of the picture's width and height, zoom of the pictures in the range of 90% to 125%, horizontal flips and shearing of 0.01 intensity.





Model accuracy on train and validation dataset

Model loss on train and validation dataset

Accuracy on test images:

Acc: 0.31670281995661603

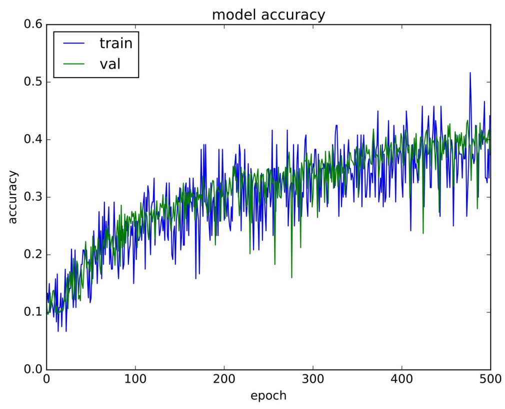
Image augmentation improved the accuracy of our model by only 0.04%. However, if we compare the graphs with previous experiment with no augmentation we can notice a higher accuracy on validation dataset than training dataset. This is caused by the fact, that model is trained on augmented twisted and turned images of lower quality. When the model is validated against real data it is easier for it to determine the correct class of the image.

Increase of the dense layer to 1024 neurons and 500 epochs:

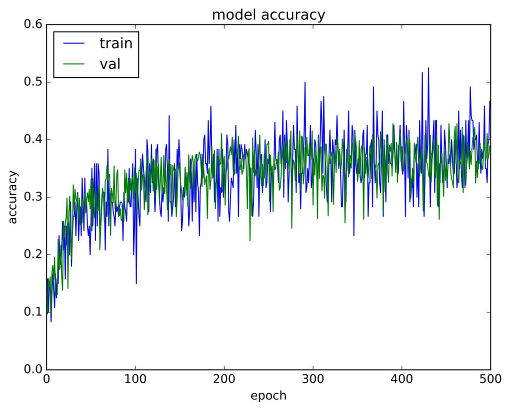
As the model is yet to reach overfitting we can try to increase the number of epochs in order to enable the model to reach this point. However, an experiment with the same network did not yield a desired improvement and ended up with accuracy cca. 0.325% without an obvious overfitting.

We attributed this to insufficiently big enough dense layers of the network, which are simply not able to store enough information to make further accuracy progress. An increase in the number of neurons in the dense layer from 128 to 1024 proved this point. We also decided to also try different activation function in order to speed up the training as the 500 epoch experiment time is approaching 2 hours.

We are presenting an accuracy graph on the validation and training dataset for 500 epoch experiment with single 1024 dense neural network between softmax classifier and CNN networks. The same experiment with exponential linear unit instead of rectified linear unit is shown on the right side.



Model with **relu** activation function accuracy on train and validation dataset



Model with **elu** activation function accuracy on train and validation dataset

```
Class  airport_inside accuracy:  0.760869
Class  bar accuracy:  0.282608695652173
Class  bedroom accuracy:  0.6086956521
Class  casino accuracy:  0.543478260869
Class  inside_subway accuracy:  0.5
Class  kitchen accuracy:  0.23913043478
Class  livingroom accuracy:  0.10869565
Class  restaurant accuracy:  0.152173913
Class  subway accuracy:  0.34782608695
Class  warehouse accuracy:  0.28260869
Acc:  0.3839479392624729
```

```
Class  airport_inside accuracy:  0.695652
Class  bar accuracy:  0.260869565217391
Class  bedroom accuracy:  0.7391304347
Class  casino accuracy:  0.347826086956
Class  inside_subway accuracy:  0.47826
Class  kitchen accuracy:  0.30434782608
Class  livingroom accuracy:  0.10869565
Class  restaurant accuracy:  0.065217391
Class  subway accuracy:  0.43478260869
Class  warehouse accuracy:  0.52173913
Acc:  0.3969631236442516
```

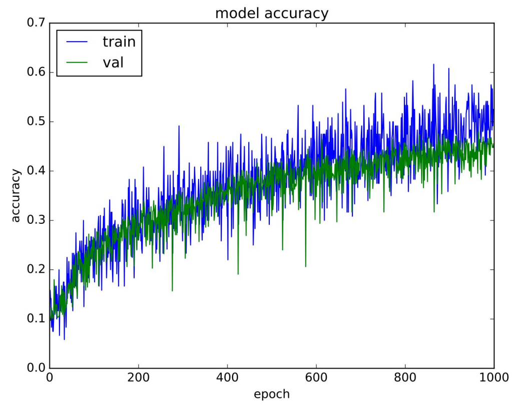
We can see a much faster learning rate of elu activation function in comparison with relu activation function as it needs only about 50 epochs to reach 30% accuracy. Elu activation function also improves overall accuracy of our model by 1.3% on the network with the same configuration. Indeed, according to some sources exponential linear units and even more advanced scaled exponential linear units yield better results than rectified linear units. [2]

In the final batch of the experiments we decided to keep using relu activation function in order to be able to compare the final results with previous ones.

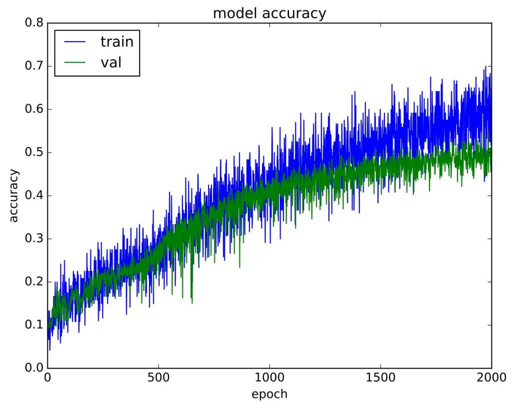
Final experiments with even more layers and epochs:

Since we had problem to allocate enough memory for the bigger dense layers than 1024 neuron layer, we were proposed to replace it with two 256 neuron dense layers. We also decreased the learning rate to a half from the default keras learning rate of 0.01 in order to tackle the spiking of the accuracy and loss metrics. We also increased the number of epochs to 1000 in order to finally achieve overfitting. The results of this experiment are on the bottom left.

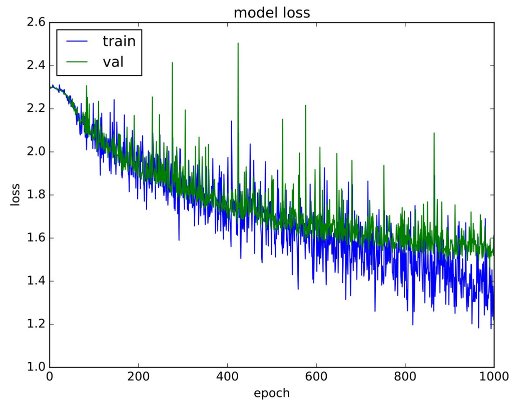
Experiment with 1000 epochs two dense 256 layers showed similar results to single 1024 dense layer, but consumes less memory. We can also notice a slight overfitting, which starts at the 800th epoch. Therefore we decided to increase the complexity of the CNN layers in order to increase the learning capabilities of the network and enable it to train for more epochs. We also decreased the learning rate to 0.0025 to decrease the spiking during the training, however it did not help to decrease the spiking by a notable difference. Our improved model has 3 convolutional layers with 64, 128 and 256 neurons, 2 dense layers each with 256 neurons and uses 2000 epochs for training. The training time of this model is approximately 7 hours.



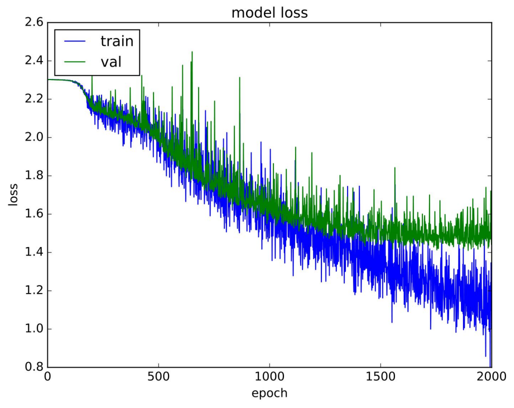
Model with 1000 epochs and 5 layers accuracy on train and validation dataset



Model with 2000 epochs and 6 layers accuracy on train and validation dataset



Model with 1000 epochs and 5 layers loss on train and validation dataset



Model with 2000 epochs and 6 layers loss on train and validation dataset

```
Class  airport_inside accuracy:  0.413043
Class  bar accuracy:  0.391304347826087
Class  bedroom accuracy:  0.8260869565
Class  casino accuracy:  0.347826086956
Class  inside_subway accuracy:  0.78260
Class  kitchen accuracy:  0.26086956521
Class  livingroom accuracy:  0.34782608
Class  restaurant accuracy:  0.239130434
Class  subway accuracy:  0.47826086956
Class  warehouse accuracy:  0.47826086
Acc:  0.45770065075921906
```

```
Class  airport_inside accuracy:  0.391304
Class  bar accuracy:  0.369565217391304
Class  bedroom accuracy:  0.760869565
Class  casino accuracy:  0.434782608695
Class  inside_subway accuracy:  0.69565
Class  kitchen accuracy:  0.52173913043
Class  livingroom accuracy:  0.28260869
Class  restaurant accuracy:  0.413043478
Class  subway accuracy:  0.3913043478
Class  warehouse accuracy:  0.67391304
Acc:  0.4945770065075922
```

Further experiments with the use of dropouts, L1 and L2 regularization did not improved the accuracy of the network. On the other hand, increasing the complexity of the network made the training time of the network too long. Therefore, in spite of overfitting of the model on the right side it has been the most model with highest test accuracy **- 49.5%**.

Conclusion:

We have presented a deep convolutional network for indoor scene recognition using MIT 67 dataset. This dataset, which was proposed to us, is not very suitable to deep learning, because it has too many classes and too little pictures per class. We have dealt with problems of various image shapes utilizing squasing, different class sample sizes using undersampling and lack of training data by the augmentation of the existing images. We have ran more than 30 experiments with various configuration and showed the continuous improvements in the accuracy by presenting the most important experiments. We have also used various activation functions and presented exponential linear unit, because other activation functions as tanh or sigmoid achieved very poor results and are not suitable for convolutional neural networks. To conclude with, we have received **49.5%** accuracy on **10** classes with relatively even accuracy distribution over various classes.

Sources:

[1] A. Quattoni, A. Torralba, Recognizing Indoor Scenes, CSAIL MIT, 1-4, available online on http://people.csail.mit.edu/torralba/publications/indoor.pdf
[2] E. Cohen, SELU — Make FNNs Great Again (SNN), 2017, available online on https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9