Cross-Modal Convolutional Neural Network

Image and Scene Recognition

Marius Marten Kästingschäfer
Maastricht University - Faculty of Psychology and Neuroscience

First Draft - November 2018
Cognitive Neuroscience - A Project supervised by Alexander Kroner and Mario Senden

1 Introduction

Since the AlexNet in 2012 won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), convolutional neural network (CNNs exploits strong local correlation in the inputs http://cs231n.github.io/convolutional-networks/) are state of the art in image processing and have dominated the challenge during the last years. During this time the models got bigger, deeper, and more complex. Further the level of precision was constantly increasing and in 2017, 29 of 38 competing teams got less than 5% wrong in the ILSVRC. This achievement on a dataset that consists of as many images as the ImageNet database implies that the networks were able to detect a wide variety of domain general features and to generate abstract representations of the real world. Whereas this indicates that the task of object detection in natural image processing is close to be universally solved this holds not true for other image classification tasks.
Different modalities such as sketches, drawings and clip art remain difficult. It would be interesting to see if the progress made on natural image detection could be transferred to these modalities. The reusability of the models for this purpose would be an advantage. Currently it reamins unclear if reusability is possible. One of the main advantage would be the wider usage of unlabeled data to obtain than fully labeled data, making reusability of networks widely applicable to many practical learning problems. To test reusability in a cross modal context this project is about training, testing and assessing a single cross-modal network. The network is compared with numerous baselines consisting of pretrained networks which are unadjusted (sequential) and another multi-input model. The pretrained unadjusted network is gonna be the AlexNet due to simplicity of the network [7] or the VGGNet [8] due to its good accuracy. Work like this is important since models become more complex and are built on top of richer algorithmic primitives. This increasing complexity requires higher reuse of networks between tasks, rather than training from scratch every time.
In short and plain language, the project 'reusing convolutional neural networks

for cross-modal scene detection and classification' is about reusing pretrained artificial neural networks to apply them on modalities they have not been trained on. I will finetune the models through hyperparameter optimization to improve their accuracy. The datasets will consist of natural images, clip art and drawings, all showing different scenes.

2 Procedure

The overall procedure consists of seven consecutive steps:
2.1 Define the problem, explicate questions and state hypothesis
2.2 Identify a way to measure success and performance of the model
2.3 Model and Data acquisition Data acqusition, model selection and acquistion, instating the convolutional base
2.4 Prepare Model and Data acquistion of weights, migrate weights, compute baseline, data augmentation, label preprocessing
2.5 Prepare the validation process N-fold cross-validation
2.6 Adjust Pretrained Convnet fine-tuning, feature extraction, training, hyperparameter optimization
2.7 Compare networks comparison with other cross-modal network, existing VGG16 baseline, computed VGG16 values

2.1 Define the problem, explicate questions and state hypothesis

Questions and the general Problem: Despite recent investment in the inner working process of convolutional neural networks some inner-working processes remain questionable.
A. How well does an CNN trained on natural images performance on classifying clip art and drawn images?
B. Are the features extracted from a wide range of natural images reusable for classifying clip art and sketches?
C. Is it possible to train a sequential model as accurate as a multi-input model with modal specific CNNs?

Hypothesis: The Hypothesis are build on assessing the literature available in the field and some prior test experience.
A. The CNN trained on natural images only will be able to classify some images correctly, but won't be able to do so on the majority of pictures. An accuracy anywhere close to the values achieved on the Place365 set by [10] would be a great success (top-5 acc. up to 85%). More might be the values attained by [2] during within model retrieval. These values vary between 19% and 31% for clip art and drawings.
B. The features in the CNN were extracted from a broad range of objects, it is assumed that they are general enough to be applied to categories of pictures

they weren't trained on. This hypothesis implicit assumes that spatial patterns in clip art and drawings are not fundamentally different from natural images.
C. The question whether more specialized networks or more generalized networks should be used might depend on the individual task. In the task at hand the hope is to fine-tune a single-input CNN to work as accurate as a more complex multi-input model.

2.2 Choice of Framework

The choice of the right framework is important since different frameworks are built in a different manner for diverging purposes. The range of available frameworks within machine learning community has grown during the last years and most of them are free to use and well suited to perform deep learning. For this project Tensorflow and Keras were choosen.
Tensorflow Tensorflow is a scalable open-source machine learning library using the programming language python. It is used for both research and production at Google. Tensorflow is able handle low-level operations such as tensor manipulations. Information can be found under https://www.tensorflow.org. The lower-level library Tensorflow is running on is called Eigen for CPU and cuDNN for GPU.
Keras Keras is a high-level neural networks API for machine learning working in a modular manner; thus allows Keras to handle several different backend engines. User friendliness is ensured by simples APIs and minimizes the number of user actions required for common use cases. Keras is maintained by an open source community under the dedicated supervision of Francois Chollet. Further information can be found under https://keras.io or here [3]. The relation between Keras and Tensorflow visualized in Figure 1.

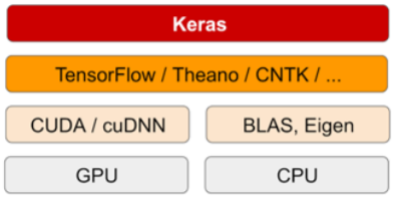

Figure 1: Deep-learning software and hardware stack. Source [3]

2.3 Identify a way to measure success and performance of the model

Accuracy During the project it is important to clarify which task the model should be optimized to do and to keep track of the inner working processes of the network. Different metrics exist to monitor success during training and testing.

On the one hand they are necessary to define a loss function and on the other hand they keep track of how well the model is performing the classification task. Accuracy is the proportion of correct results that a classifier achieves, it is a standard measure of performance and will also be applied during this project. (In our case there are only correct or incorrect classifications, false positive or true negatives are equally wrong classified.)
Different forms: R2 Value of your model, Adjusted R2 Value, RMSE Value, Confusion Matrix (clarify statistical approach and optionally add mathematical backbone) - How does the measure of success define the loss function the optimizer (binary crossesntropy)

2.4 Model and Data acquisition

Data acquisition The natural images consist of the Scene205 dataset [9] with around 2.5 million images from 205 scene categories. 100 images per category for validation set and 200 images for test set. The place205 can be downloaded freely under: http://places.csail.mit.edu/user/download.php. The compressed file of resized 256*256 images, containing train set and validation set of Places 205 has a size of 126GB. Since only around 30-50 picutres per objects will be needed some images will be deleted and around 16,000 pictures will remain. The Sketch images consist of 14,830 training and 2,050 validation sketches collected through AMT, whereby different colors indicate different objects. They were label with Amazon Mechanical Turk. The Clip art data includes 11,372 training and 1,954 validation clip art images downloaded from search engines. Both, clip art and sketches were taken from the http://projects.csail.mit.edu/cmplaces/download.html. Examples for scenes labels are bookstore, lobby and harbor. Pictures from the datasets are shown in Figure 2.

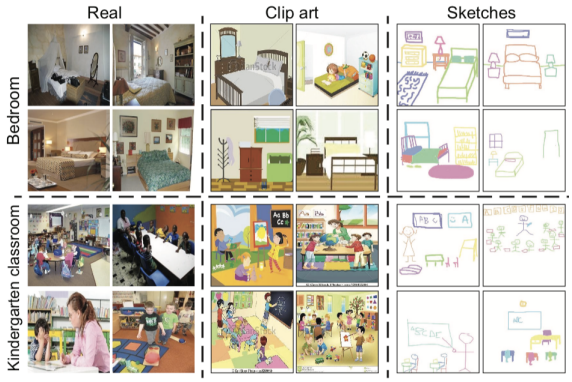

Figure 2: Examples of the cross-modal dataset showing natural images, clip art and sketches or drawings. Source [2]

Model selection and acquisition Using a pretrained network is typically much faster and easier than training a network from scratch. Further it is

especially important to use pretrained networks to assess the research question in an experimental manner. The decision for the pretrained unadjusted network is made between two networks. The AlexNet due to simplicity of the network [7] and the VGGNet [8] due to its good accuracy. The decision which network is chosen is based on Figure 3 from [10], the networks were measured on the Place365 image set (consisting of natural images only). For this project the VGG16 was chosen, mainly because it is one of the best established networks available and due to the fact that architecture is less complex (one stream only).

	Validation Set of Places365		Test Set of Places365	
	Top-1 acc.	Top-5 acc.	Top-1 acc.	Top-5 acc.
Places365-AlexNet	53.17%	82.89%	53.31%	82.75%
Places365-GoogLeNet	53.63%	83.88%	53.59%	84.01%
Places365-VGG	**55.24%**	84.91%	**55.19%**	85.01%
Places365-ResNet	54.74%	**85.08%**	54.65%	**85.07%**

Figure 3: Comparison of different networks on Places365 consisting of natural images only. Important to concider that the values are obtained on a dataset with 160 more images than the Place205 used for this project. Source [10]

The VGG16 architecture shown in Figure 4 is prepacked with Keras and can be importet from keras.applications module (https://github.com/fchollet/deep-learning-models/blob/master/vgg16.py). The pretrained weights will then be implemented. The network contain 13 convolutional layers and 5 maxpooling layers. Relu is applied after every convolutional layer. The network has 14.714.688 million parametersa and was first introduced in [8]. Weights will be acquired in step 2.4 migrate weights and implemented and adjusted into the network in step 2.6.

Instantiating the VGG16 convolutional base

```
from keras.applications import VGG16
conv_base = VGG16(weights='Imagenet365',
                  include_top=False)
```

Model summary

```
conv_base.summary()
```

The command 'weights' specifies the weights the model will be initialized with, here the pretrained weights will be included. 'Include top' refers to including the densely connected classifier on top of the network. Since it is intended to retrain this part the connected classifier don't need to be included. Input shape is an optional argument, the network will be able to process all inputs if it is not passed. The final feature map has shape (4, 4, 512).

2.5 Prepare the evaluating process

Training, validation, and test sets To reproducible evaluate the models the data will be split into a training set, a validation set, and a test set. The training

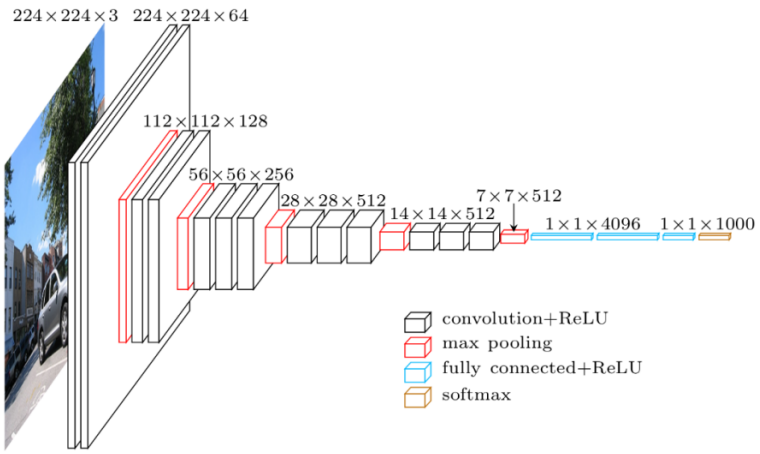

Figure 4: Macroarchitecture of the VGG16. Source: https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/

set will yield 70%, the validation set 20% and the test set 10% of the total data. Evaluating the models on the same data as they were trained would quickly lead to overfitting. Instead the model is trained on the training data for learning the parameters. The validation data is then used for deciding which settings of the meta parameters work best. In the end the test data is used to get a final, unbiased estimate of how well the network works. This is normally worse than in the other two sets. Within the data each set it will be hold an equal amount of data (in a random order) to ensure as much independence as possible.

Split data

```
import os, shutil

#uncompressed original dataset
original_dataset_dir = '/Users/'

#smaller dataset
base_dir = '/Users/'
os.mkdir(base_dir)

#directories for training, validation, test split
train_dir = os.path.join(base_dir, 'train')
os.mkdir(train_dir)
validation_dir = os.path.join(base_dir, 'validation')
os.mkdir(validation_dir)
test_dir = os.path.join(base_dir, 'test')
```

```
os.mkdir(test_dir)

#directory with training, validation and test pictures
..
```

2.6 Prepare Model and Data

Acquisition of weights The weights used are trained in Python2.7 + PyTorch 0.2 and can be acquired here https://github.com/CSAILVision/places365 provided by [10]. The weights were trained on the train set of Places365-Standard consisting of 1.8 million images from 365 scene categories. As shown in 2 the VGG16 has a top-1 acc. of 55.24% and a top-5 acc. of 84.91% on the places365. Comparable results were obtained in the places205 (used in this project) namely a top-1 acc. of 58.90% and a top-5 acc. of 87.70%. Some qualitative prediction results using the VGG16-Places365 can be seen in Figure 5.

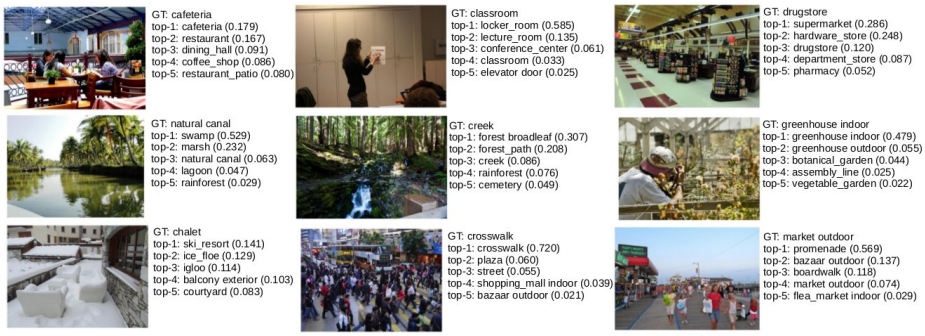

Figure 5: Predictions using the VGG16-Places365 (only natural images)

Migrate weights The acquired weights are optimized for a use in PyTorch. To make them accessible for the Tensorflow framework and accompanied Keras they need to be converted. The migration from Torch to Tensorflor is done with MMdnn a tool from Microsoft to inter-operate among different deep learning frameworks (https://github.com/Microsoft/MMdnn). MMdnn stands for model management and "dnn" is an acronym for the deep neural network. The tool converts the model by producing an intermediate representation format.
Compute baseline There are different sources of information to compare the final accuracy measures with, one of them is to compute a baseline with the unadjusted VGG16. For this purpose the pretrained VGG16 will be used to classify the clip art and drawings into the existing categories. During this process some of the place365 labels will muted. After this the VGG16 architecture (conv base) will be completed by adding Dense layers on top, and running the network end to end on the different input data. Since every images goes through the convolutional base it is possible to use data augmentation.

Data preprocessing Before starting to work on them the JPEG files need to be preprocessed into floating-point tensors, also called vecorization. For this purpose the files will be read, the JPEG content will be decoded to RGB grids of pixels and then converted into floating-point tensors. Those tensors will be rescale to values between 0 and 1. Using Keras keras.preprocessing.image especially the ImageDataGenerator makes this processing handy. The output of one of these generator yields batches of 150x150 RGB images (shape (20, 150, 150, 3)) and binary labels.

Read images from directories

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(150, 150)
        batch_size=20,
        class_mode='binary')
validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
```

Data augmentation To fight overfitting, caused by having too few samples to learn from data, data augmentation could be used. Data augmentation generates more training data from existing training samples by augmenting existing samples via a number of different transformations. This makes the training computationally more expensive but will hopefully improve performance significantly. Additionally cropping the data is not necessary since the drawings and the clip art pictures were made for the image classification, so there structure is well suited for the task.

Augmentation

```
datagen = ImageDataGenerator(
      rotation_range=40,            #randomly rotation
      width_shift_range=0.2,        #changes vertical range
      height_shift_range=0.2,
      shear_range=0.2,              #shear transformation
      zoom_range=0.2,               #zooming inside the picture
      horizontal_flip=True,         #flipping image horizontally
      fill_mode='nearest')          #filling in newly created pixels

test_datagen = ImageDataGenerator(rescale=1./255)
#no augmentation of validation data
```

Label preprocessing A category in a classification problem is called a class, the class associated with a certain sample is called a label. There are 205 labels, one for each category. The labels of the sketches and the drawings are indicated by the folder they are saved in.
Label Preprocessing

```
Code
```

2.7 Adjusting Pretrained Convnet

The decision for how to train the pretrained network is made between fine-tuning and feature extraction. A comprehensive explanation of **fine-tuning** is that all weights would be changed during training on the (partly) new task and during **feature extraction** only the weights of the newly added last layer would change. The difference is shown in 6. For the task and project, feature extraction will be used due to its easier implementation and for being less computational expensive. The model adjusted with feature extraction will be trained end to end with a frozen convolutional base. Further information can be found under: https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html.

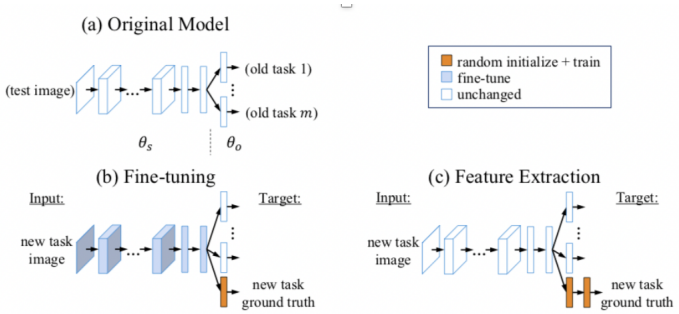

Figure 6: Differences in using pretrained models. Source [3]

Up to now the network was build and the converted weights were implemented. Now an classifier will be added to the pretrained convolutional base and only this part will be optimized furing consequtive training.
Add classifier The classifier will consist of a sequence of two Dense layers, which are densely connected. Dense layers learn global patterns in their input feature space; whereas convolution layers learn local patterns. Those fully connected layers comprise a relu layer and a sigmoid layer. The classifier has 2 million additional parameters.

Adding a densely connected classifier on top of the convolutional base

```
from keras import models
from keras import layers
```

```
model = models.Sequential()         #forward only
model.add(conv_base)                #pretrained network
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Freeze convolutional base Before compiling and training the model it is necessary to freeze the convolutional bases trainable attributes. Without freezing the previously learned representation would be modified during training. This is particularly running into problems since the Dense layer is randomly initialized and very large weight updates would be propagated through the network.

Freeze convolutional base

```
from keras import optimizers

train_generator = train_datagen.flow_from_directory(
        train_dir,                      #target directory
        target_size=(150, 150),         #resize all images to 150x150
        batch_size=20,
        class_mode='binary')            #binary labels for binary crossentropy

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
```

Define loss function and optimizer The loss function (sometimes also called objective function) is the quantity that will be minimized during the training. It enables to measure the models performance on the training data and enables the model to steer itself in the right direction. This makes the choice of the right loss function essential. In our case the categorical crossentropy loss function is used. Crossentropy is the number of bits we'll need if we encode symbols from y using the wrong tool $\hat{y}$, so it takes the distance to the goal into account. Since we have multi-class classification where each example belongs to a single class we use categorical cross entropy instead of binary cross-entropy is for multi-label classifications. The optimizer is a specific variant of stochastic gradient descent (SGD), it defines how the network will be updated based on the loss function. The relationship between loss function and optimizer is shown in Figure 7. As optimizer RMSprop will be used, it divides the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. It defines the learning rate to be lr=2e-5.

Define loss function

```
model.compile(loss='binary_crossentropy',
```

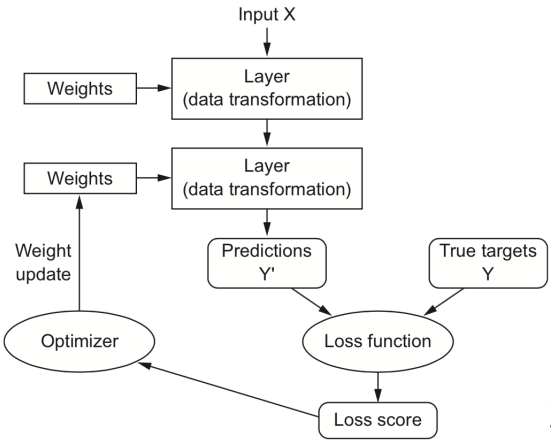

Figure 7: Association between different blocks of the network. The loss function measures the quality of the network's output and the loss function score is consecutive used as a feedback signal to adjust the weights [3]

```
optimizer=optimizers.RMSprop(lr=2e-5),
metrics=['acc'])
```

History - fit.model Model.fit() returns a history object. This object has a member history, which is a dictionary containing data about everything that happened during training. Those results will be plotted in the next step. It also defines the number of epochs to train and the size of the batches used.

History

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50)
```

Callback Callback is an object (a class instance implementing specific methods) that is passed to the model in the call to fit and that is called by the model at various points during training. This way the callback has access to the model and its performance. A callback is able to take actions such as interrupting training (early stopping), saving a model, loading a different weight set or alter hyperparameters during trainig. For visualisations with the TensorBoard (an integrated metrics montitor of Tensorflow) Tensorbaord callbacks will be used.
Pause training To make some first test running and training the network on a local system will be needed. To ensure that the advancements were saved and to interrupt training it make sense to pause the training. An example for

how this is done can be find here: https://www.kaggle.com/morenoh149/keras-continue-training.
Plot results Displaying curves with loss metrics and accuracy changes during and after training is always useful. It gives insides into the inner working processes of the model and enables us to detect overfitting.

```
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```

Save model

```
model.save('adjusted_alexnet_1')
```

Evaluate model on test data:

```
est_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
print('test acc:', test_acc)
```

Hyperparameter optimization An architecture-level parameters is called hyperparameter. Choosing those parameters like learning rate of the optimizer, numbers of layers, adding dropout [5] or regularization method are often somewhat arbitrary. This is why data engineers often consider deep learning model building to be more like an art than a science. This makes it compulsory to repeatedly modify the model, train it and evaluate on the validation data again. The goal is to achieve as much generalization as possible or to minimize overfitting.
Cloud computing Multimodal training data yields the danger of strong interference effects which lead to slow learning and poor generalization. Interference

can be minimized by finding the optimal hyperparamaters that best suit the model. This requires reiterated tests with different hyperparameters, due to being computational expensive this task is well suited for cloud computing. Resources for cloud computing can be found here https://console.cloud.google.com/compute/.

2.8 Compare networks

The purpose of this experiment is to get as much information about how well a model performs as possible. To compare the results obtained the outcomes of similar other models will be consulted. There are three different sources of information to compare the final accuracy measures with.

Comparison Cross-modal scene network First the results of the cross-modal scene network from [2], they use a high-level representation that is shared across all modalities. The values can be seen in in Figure 8. There only NAT (natural images), CLP (clip art) and SPT (sketches) are important. The commission was to retrieve images from the same modality (within modality retrieval), meaning the query and the target were the same modality. The values vary between 19% and 31% for clip art and drawings. The figure 9 shows mean average precision of the model.

Not shown here are values for precision at top 10 retrieval of images across modalities using fc7 features. These values are: Query NAT - Target CLP (18.2%) and Quesry NAT - Target SPT (16.6%), Query CLP - Target Nat (24.8%) and Quesry CLP - Target SPT (10.2%), Query SPT - Target NAT (21.1%) and Query SPT - Target CLP (9.0%). All values are taken from [1]. From this values it can already be obtained that it will be difficult to achieve a high accuracy. The values are only indirectly comparable with the values of this project, but they are still helpful to assess a baseline (since they used the same data sets).

Within Modality Retrieval	NAT	CLP	SPT	LDR	DSC	Mean
BL-Ind	19.3	31.7	83.0	18.1	11.1	32.6
BL-ShFinal	18.2	22.0	81.2	13.8	29.8	33.0
BL-ShAll	18.4	26.7	82.7	16.6	11.1	31.1
A: Tune	19.0	23.9	74.2	13.7	36.3	33.4
A: Tune (Free)	19.4	22.9	85.0	13.5	34.2	**35.0**
B: StatReg (Gaussian)	19.4	31.1	84.0	17.3	11.1	32.6
B: StatReg (GMM)	19.3	31.1	82.5	16.7	13.5	32.6
C: Tune + B: StatReg (GMM)	20.2	22.5	82.2	13.1	37.0	**35.0**

Figure 8: Within Modal Retrieval mAPs - mean average precision (mAP) for retrieving images within the same modality using fc7 features. Source [2]

Existing VGG16 baseline The second comparison can be done with the networks already mentioned under model selection. Top-5 accuracy is up to 85% and top-1 accuracy is 55%. figure 3 showed the results in detail. It needs

Cross Modal Retrieval	Query	NAT				CLP				SPT			
	Target	CLP	SPT	LDR	DSC	NAT	SPT	LDR	DSC	NAT	CLP	LDR	DSC
BL-Individual		17.9	11.9	10.0	1.3	12.2	10.3	9.2	1.3	7.0	9.1	**5.2**	1.1
BL-Shared-Upper-Scratch		7.0	7.8	4.1	10.9	5.5	5.0	3.2	9.2	5.2	4.5	2.7	8.9
BL-Shared-Upper		10.4	12.4	4.5	14.6	9.1	7.2	3.7	10.1	6.8	5.5	3.0	8.9
Subsp. Align. [12] + PlacesNet		9.4	0.8	3.1	-	8.2	0.9	2.4	-	0.7	0.9	0.9	-
Subsp. Align. [12] + PlacesNet Finetune		16.3	10.2	10.4	1.2	14.1	9.4	**10.5**	1.5	8.6	**9.4**	5.1	1.2
A: Tune		13.3	11.3	6.7	21.9	10.1	8.5	5.7	15.8	6.3	4.8	3.4	11.4
A: Tune (Free)		14.0	16.0	7.9	20.6	9.6	8.1	4.7	14.8	**11.3**	8.0	**5.2**	18.0
B: StatReg (Gaussian)		17.3	11.9	10.1	1.6	12.6	8.9	9.7	1.3	6.6	8.6	4.9	1.4
B: StatReg (GMM)		**18.2**	11.3	**10.5**	1.2	**14.5**	**10.7**	10.1	1.2	7.0	7.9	4.9	1.2
C: Tune + StatReg (GMM)		13.2	**16.9**	7.2	**24.5**	10.9	10.4	5.7	**16.5**	10.1	8.3	5.0	**18.8**

Figure 9: Cross-Modal Retrieval mAP - mean average precision (mAP) on retrieving images across modalities using fc7 features. Each column shows a different query-target pair. On the far right, the average over all pairs is shown. For comparison, chance obtains 0.73 mAP. Best performances in each column are highlighted as boldSource [1].

to be taken into account that these values were obtained using a extremely fine-tuned unimodal network using only natural images.

Computed VGG16 baseline The third baseline consists of the self computed baseline under 2.6 prepare model and data - compute baseline. Compute untrained baseline (on the rest of the images) and Split data. Last it should be mentioned that the chance of retrieving the correct label by chance is 1/205 = 0.49%. To assess how well the model is doing the results obtained with be compared with all three (four) baselines.

Additional parameters to compare Possible questions that could be taking into consideration while assessing the model: On what was the model trained (details on CPU)? How long was the model trained? On how many examples? How many steps? Differences in learning rate and optimizer. Does

3 Optional Procedure

The Marble dates and deadlines are the following:

1. Deadline submission Intro+Methods Thesis — February 2019
2. Research data collection until — March 2019
3. Writing empirical bachelor thesis until — April 2019
4. Deadline submission complete thesis — mid April 2019
5. Poster presentations annual Marble conference — July 2019

Getting the most out of my model, using state-of-the-art-deep-learning techniques to improve performance.

Batch Normalization: Normalization is generally done to make the inputs of an model more similar to each other (often done by centering the data on 0).

Batch normalization does a normalization after each layer even when mean and variance change over time [6].

normalization after each layer

Advanced Overfitting: more data, smaller model (architecture, early stopping, weight-decay / penalize larger weights, noise) or change metaparameters (hidden units etc - here it is important to adjust metaparameters to the validation set and not to the training data, only test data gives unbiased estimate of how well network works), average different models, work with bayesian methods **Depthwise separable convolution**: performs a spatial convolution on each channel of its input independent, before mixing output channels via pointwise convolution

3.1 Discussion

unsupervised pretraining and fine-tuning with backpropagation works very efficiently and could be an alternative method (since it increases performance and quality of the extracted features) - Phone recognition on the TIMIT benchmark (Mohamed, Dahl, and Hinton, 2009 and 2012) training the same model to do several loosely connected tasks at the same time results in a model that's better at each task

The quality of the dataset is generated by humans but it is not validated by humans afterwards. In some cases it remains doubtful if the images would be classified correctly by humans. This makes the task eminently hard.

Figure 10: An example taken from the sketches data set of an outdoor swimming pool. Source: http://projects.csail.mit.edu/cmplaces/

3.2 Mathematical Framework

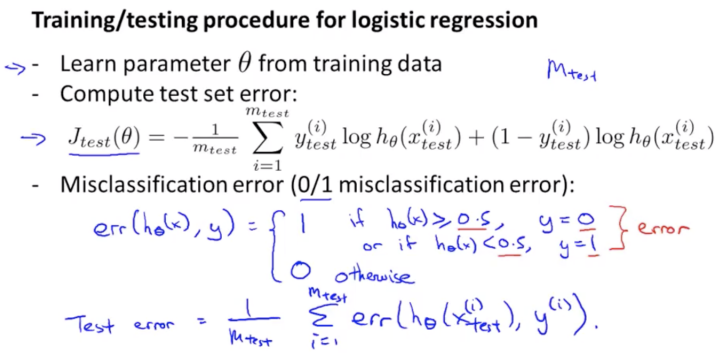

Figure 11: Error calculation. Source Andrew Ng - Machine learning

3.3 Visualizations

Model Visualiazation https://github.com/Microsoft/MMdnn The MMdnn model visualizer and submit your IR json file to visualize your model. In order to run the commands below, you will need to install requests, keras, and TensorFlow using your favorite package manager. Network Structure - use TensorFlow computation graph for the visualization of the CNN, to illustrate each of the layers and computation done in between each layer.

Hidden Unit Visualizations If possible visualizations of the project will be done. It would be interesting to see: How do intermediate actions look like? How does generalization changes with improvement of accuracy? How does abstraction increases within the network?

How does differences in accuracy change the representation within the layers?

Use Keras (Visualizing intermediate convnet outputs (intermediate activation's, Visualizing convnets filters, Visualizing heatmaps of class activation in an image)

Using Tensorboard to visualize

Further information (https://distill.pub/2017/feature-visualization/)

3.4 Biological processes

Animals and especially humans are able to leverage knowledge and experiences independently of the modality they perceive it in. A similar capability in machines would be an outstanding achievement. It could be of value to have a look at biological processing underlying vision (object detection, shape and texture differentiation, finding edges, discriminate figure from ground, , color, resolving interference and combining multiple cues especially during motion)

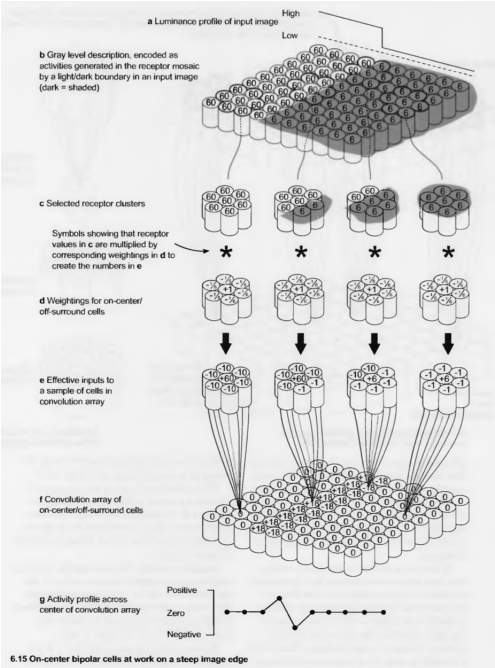

Figure 12: Example of On-center bipolar cells for detecting edges. Source [4]

References

[1] Yusuf Aytar, Lluis Castrejon, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Cross-modal scene networks. *CoRR*, abs/1610.09003, 2016.

[2] Lluis Castrejon, Yusuf Aytar, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Learning aligned cross-modal representations from weakly aligned data. 2016.

[3] François Chollet. *Deep Learning with Python*. Manning Publications, 2017.

[4] John P. Frisby and James V. Stone. *Seeing: The Computational Approach to Biological Vision*. MIT Press, 2009.

[5] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. 2012.

[6] Sergey Ioffe and Christiano Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. neural information processing systems. 2012.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[9] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. 2014.

[10] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.