UNIVERSITEIT VAN AMSTERDAM

# A Deep Learning Ensemble approach to the Yelp Restaurant Classification Challenge

Joris Baan
10576681

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

*Supervisors*
Dr Estratios Gavves
Noureldien Hussein

Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 24th, 2016

# Contents

**Abstract**

The goal of the YRCC is to build a model that automatically labels restaurants with multiple categories based on user-submitted photos. The aim of this thesis is to use convolutional neural networks and experiment with different deep learning architectures in order to extract and classify features that contribute to the highest F1-score in the YRCC. The best performing model is a multi-model ensemble that fuses the predictions of three neural network trained on features extracted from VGGNet16 pre-trained on Imagenet, Places or Food101. The optimal fusion method is to average the probabilities of the neural networks and results in an F1-sample score of 0.82 on the test set and the 25th rank on the Kaggle leaderboards.

# 1 Introduction

Yelp is a popular social networking website that allows users to share reviews and information about businesses. In addition to written reviews, users can upload photos and categorize businesses. A great deal of businesses, however, are only partially categorized or not at all. The Yelp Restaurant Classification Challenge (YRCC) is a challenge hosted by a data science community known as Kaggle in collaboration with Yelp. The goal of the YRCC is to build a model that automatically labels restaurants with multiple categories based on user-submitted photos. A restaurant can be associated with one or more of the following nine labels:

```
0: good_for_lunch
1: good_for_dinner
2: takes_reservations
3: outdoor_seating
4: restaurant_is_expensive
5: has_alcohol
6: has_table_service
7: ambience_is_classy
8: good_for_kids
```

This challenge has two interesting and unconventional technical aspects to it. The first is the binary multi-label classification aspect, which means that every restaurant corresponds to a set of labels instead of one individual label. The second aspect is the multiple-instance nature of the challenge. In conventional image classification tasks, every image has a class or labels associated with it. In the YRCC, only restaurants are labeled which means that there is no information about the images except for their correspondence to a specific restaurant. This increases the difficulty of developing predictive models based on the images.

The aim of this thesis is to experiment with different deep learning architectures in order to extract and classify features that contribute to the highest F1-score in the YRCC. The first experiments will consist of utilizing different convolutional neural network (CNN) architectures, which are explained in detail in section 2, as well as determining the optimal layers to extract features from. The effects of combining features from different layers will also be investigated. Subsequently, the best CNN architecture will be trained on thee different datasets: Imagenet, Food-101 and MIT's places. These models will be compared quantitatively in terms of overall F1-score and F1-score per label, as well as qualitatively by examining the restaurants of which the models are most confident to investigate what kind of images contribute to a certain label. The different models will then be combined by *early fusion* and *late fusion* [18]. Since the models are trained on different datasets they are expected to extract complimentary features, and

therefore combining them into a multi-model ensemble is expected to achieve superior results. Finally, a suitable classification method will be established, although the focus of this thesis is more on improving feature extraction rather than classification.

It is especially interesting to deploy networks trained on different datasets because the images are all user-uploaded and therefore very diverse. Common images are of food, the restaurant's interior and exterior, the menu, people, and even bathrooms. This means that images contain both objects, scenes and different types of food. Intuitively, a network trained for scene recognition should be better suited for predicting whether a restaurant has outdoor seating, whereas a food classifier should be better at predicting whether a restaurant is good for lunch. A multi-model ensemble is thus expected to combine the strengths of the various models and achieve the highest predictive power. The following section provides a theoretical framework regarding technical and historical aspects of convolutional neural networks, CNNs as feature extractors, multi-label classification and a discussion of recent work on CNN ensembles.

# 2 Relevant work

Convolutional neural networks are deep neural networks that are based on the hierarchical organization of the visual cortex. They are currently very popular in the field of Computer Vision and widely used for image classification and object recognition tasks. When they were first implemented around 1980, however, the backpropagation algorithm did not exist yet which made early implementations very inefficient. The first breakthrough came in 1998 when a small CNN was used to read bank cheques.

## 2.1 Gradient-based learning applied to document recognition (LeNet-5)

LeNet-5 was a CNN that could recognize handwritten digits by using backpropagation in a convolutional feedforward net. It was developed by LeCun et al. [9] and was the first successful example of a CNN. One of the major reasons that this CNN worked so well was due to the use of backpropagation algorithm that computed the parameter updates of the filters by calculating the gradient for each neuron with respect to the output layer in order to minimize the loss. As shown in figure 1, LeNet-5 takes a 32x32 image as input and is comprised of several layers that in turn perform convolutions and pooling. The convolution layer consists of multiple layers of neurons, known as feature maps. The first layer of LeNet-5 has six feature maps, the second convolutional layer has sixteen and the third 120. In these feature maps all neurons share the same parameters and compute the dot product between a 5x5 area of the image and a matrix with shared parameters, known as a filter. This dot product is subsequently passed through a Sigmoid function. The parameters of a filter are learned through backpropagation and are the core of feature extraction using CNNs.

After a convolution step, which created X feature maps by sliding X different 5x5 matrices over the image, pooling is conducted. The goal of pooling is to reduce the network's sensitivity to shifts and distortions of features. The exact location of features is irrelevant and might even be harmful because the position of the feature might differ even though it still represents the same entity. Pooling reduces the size of the feature maps by computing the mean over local areas of 2x2 and passing it through a Sigmoid function, thus reducing sensitivity to shifts and distortions. After three convolutional layers and two sampling layers, which can be seen as the

feature extraction layers, LeNet-5 uses a fully connected layer and a Gaussian connected layer for classification, achieving an error rate of merely 0.95%. With LeNet-5, LeCun et al. showed that CNNs eliminated the need to create feature extractors by hand, because CNNs allow for automated feature learning from examples, therefore setting the standard for CNN architectures.

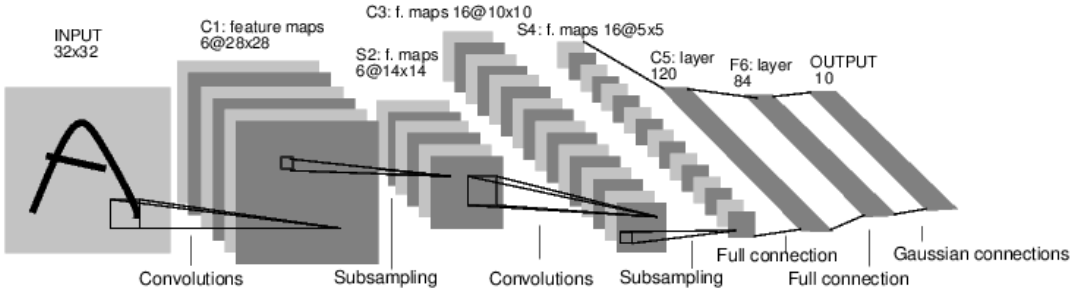



Figure 1: A high level overview of the LeNet-5 architecture [9]

## 2.2 ImageNet classification with deep convolutional neural networks (AlexNet)

The next breakthrough in convolutional neural networks came in 2012 when Krizhevsky et al. [8] won the Large Scale Visual Recognition Challenge (ILSVRC)[13] . This challenge serves as a benchmark in object category classification and detection using the Imagenet[4] database. This is a large database of annotated images organized according to the WordNet hierarchy. The winning system was a *Deep* convolutional network known as AlexNet, which beat the competition with a top-5 test error rate of 15.3% compared the 26.2% achieved by the number two. Even though there was a twelve year gap between LeNet-5 and AlexNet, the overall structure was very similar. There were three major differences between them, however.

The first difference was that instead of using Sigmoid units for the activation function, Rectified Linear units (ReLU) were used in every hidden layer allowing for more efficient training and expressive power. The second improvement was the augmentation of the data. Instead of solely using the input image, AlexNet trained on random patches of 224x224 and left-right reflections of those patches, yielding a larger train set and assisting with translation invariance. At test time, the output of ten different patches (the four corners, the middle, and their reflections) were combined into a single opinion. The third improvement was the use of dropout to regularize the weights in the globally connected layers in order to prevent overfitting. The basic concept of dropout is that for each training example half the neurons in a layer are randomly omitted, preventing them from being dependent on the other half's presence. Thus, they are forced to learn expressive features by themselves and stop cooperating with other hidden units. Finally, Krizhevsky et al. used a combination of multiple CNNs and their predictions, which improved the accuracy even more and demonstrated that an ensemble of multiple CNNs bears better results than using a single CNN.

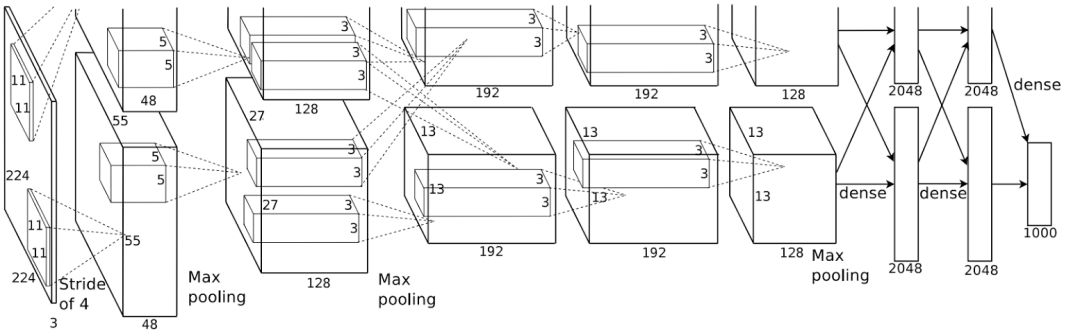



Figure 2: A high level overview of the Alexnet architecture [8]

## 2.3 Very deep convolutional networks for large-scale image recognition (VGGNet)

GoogLeNet [16] was the winner of the ILSVRC 2014 and had impressive results, both parameter-wise and error-wise, but it had a pretty complex structure. In contrast, the second place winner, VGGNet by Simonyan et al [15], was much simpler. This CNN was very interesting because it showed that the depth of a CNN is crucial for performance. Instead of tweaking the filter size, stride, zero padding, or adding new smaller networks in the network such as GoogLeNet, a 3x3 convolution kernel and 2x2 pooling kernel was used from beginning to end. Only the number of convolutional layers, in other words the depth of the network, was experimented with. Even though the error rate of the VGGNet was slightly higher than GoogLeNet's, VGGNet seemed to be better at transfer learning tasks, which makes it an interesting network for this research.

The architecture of VGGNet is very similar AlexNet's architecture. To quickly summarize: a 224x224 RGB mean-subtracted image is zero-padded which preserves the resolution, after which the image is fed through a stack of convolutional layers (3x3 kernel, stride one) and some max-pooling layers (2x2 kernel, stride two). The activation function of the neurons is ReLU. The number of convolutional layers is varied from eight to sixteen in five different networks A through E.

Two consecutive convolutional layers have an effective receptive field of 5x5 with the advantage of having an extra layer of ReLU which allows for a more discriminative decision function. Furthermore, it decreases the amount of parameters which has a regularizing effect. These effects become stronger with every weighted layer added and are arguments in favor of increased depth in neural networks. GoogLeNet, just like VGGNet, achieved good results with a very deep network, but achieved this using a much more complex network topology which is harder to understand and exploit. VGGNet demonstrated that merely increasing the depth along with constant, small convolution filters positively influences the network's performance, ultimately achieving a 7.3% error. Recently, CNN architectures that are even deeper and perform better have emerged, for example ResNet [6], but these networks are so computationally expensive that they are not applicable to this research.

## 2.4 Pre-trained convolutional neural networks as feature extractors

Convolutional neural networks pre-trained on Imagenet are often used as feature extractors in image recognition tasks. Thus, it is not only important investigate at the best architecture, but also at what exactly happens inside such an architecture. This is important because features can be extracted from any layer, therefore understanding the difference between layers can assist in extracting optimal features. Yosinki et al. [21] argue that the first layer learns very general features that primarily represent Gabor filters and color blobs, whereas the final layer learns very task-specific features that greatly depend on the dataset and task. Their research is concerned with examining how and where the transition of general to specific features materializes. This led to the discovery that there exists a fragile co-adaptation between successive layers mainly in the middle of the network, which makes extracting features from those layers less than ideal. A second discovery is that there is a steady decrease in generality of features as the layers increase in depth.

Zeiler et al. [22] also attempted to gain a better understanding of features extracted from CNNs and observed that features generated by CNNs are actually meaningful patterns with increasing invariance and class discrimination as the layers get deeper. Moreover, they verified that features get more powerful and discriminative as they are extracted from deeper (later) layers. This supports the common claim in deep learning that features should be extracted from the fully connected layers, which is also more efficient from a computational point of view since the features from the fully connected layers are much smaller in size than features from the convolutional layers.

Generally when pre-trained CNNs are used as feature extractors they have been trained on Imagenet. However, there are datasets from other domains created for the purpose of training CNNs as well. Two interesting datasets for this research are MIT's Places [24] and Food-101 [2]. Places was developed for the purpose of scene recognition and features 205 different scenes each containing at least 5000 images. Food-101 is a dataset containing 101 different types of food where each type is represented by 1000 images and enabling for food recognition and classification.

## 2.5 Convolutional neural network ensembles

A reliable approach to improving the performance of CNNs is to train multiple independent models and combine their results [18]. This method is known as ensemble learning and is based on the assumption that each model in the ensemble produces different types of errors and performs on different classes. By combining the CNNs a more general model should emerge that performs better on all classes. Wei et al. [18] proposed two methods of fusing the CNNs: early fusion and late fusion. Early fusion combines the features extracted by different models and subsequently classifies using the combined features, whereas late fusion combines the predictions of the individual models after classification. There are different types of late fusion methods, one of which is termed bootstrap aggregating or bagging [3]. Bagging combines the predictions of multiple classifiers by a voting procedure and involves training the classifiers on different randomly selected parts of the training data in order to promote model variance. Another late fusion method is known as stacked generalization, or stacking, and consists of training another learning algorithm to combine the scores of the models in the learning ensemble. A commonly used algorithm for this is logistic regression [20]. Wei et al. achieved superior results by combining five different CNNs trained and fine-tuned on different datasets. Virtually all past ILSVRC winners used an ensemble of slightly different CNNs instead of a single model as well [6, 8, 15, 16].

### 2.6 Multi-label classification

Now that a theoretical framework for deep learning has been established, the multi-label aspect of the YRCC will be addressed. Multi-label classification is a classification problem where multiple target labels must be assigned to each instance. There are two ways to approach a multi-label problem: problem transformation, which transforms the problem into a single-label problem, and algorithm adaptation, which uses algorithms developed specifically for multi-label classification or regression [23]. A problem transformation method that is often used as baseline is known as binary relevance and involves training one classifier for every label. The advantages of binary relevance are the conceptual and computational simplicity and the fact that it handles irregular labelling well, contrary to the label powerset transformation approach where each label combination is treated as a single class. The problem, however, is that relations between labels are not captured. A relatively new multi-label classification method based on binary relevance, known as Classifier Chains, aims to solve this problem by creating a new dataset for every label and adding the remaining labels as features to the feature vectors. During test time the classifiers pass label information to one another through the feature space, effectively forming classifier chains that do capture relations between labels [12]. Because the order in the chain matters, ensembles of classifier chains can be used to achieve a higher score.

Opposed to the problem transformation methods, a state-of-the-art algorithm transformation method is the use of multi-label neural networks. The greatest advantage of neural networks is that they capture inter-label dependencies [11].

## 3 Research method

### 3.1 Data

The data consisted of a labeled train set with 2000 restaurants and roughly 230.000 images along with an unlabeled test set with 10.000 restaurants and approximately 1.2 million images. The images depicted a wide variety of objects and scenes, but according to an engineering blog on Yelp [19] there were five main categories to be distinguished: food, drinks, interior, exterior and menus. One representative photo for each category was selected from the dataset and is shown in figure 3. Even though the test set was unlabeled, predictions could be evaluated by submitting a file with restaurant IDs and predicted labels to Kaggle. Kaggle subsequently calculated the Mean F1-score (discussed in section 4.1 and the corresponding rank on the leaderboards based on that score. A great deal of the experiments were done using just the train set because it allowed for faster testing cycles and a more detailed analysis and evaluation in terms of performance per label. This was more valuable than merely an overall F1-score since it provided insights into the strengths and weaknesses of a model on different types of images and labels. Although the dataset occurred as a great deal of data to train on, in fact there were only 2000 labeled businesses, which is actually quite scarce for deep learning standards.

### 3.2 Technical details

**DAS-4** All experiments were done on DAS-4 [1] using NVIDIA's GTX Titan X graphics card. DAS-4 is the fourth generation of a medium-scale supercomputer project from the Dutch Advanced School for Computing and Imaging (ASCI).


(a) Food (b) Drinks (c) Interior (d) Exterior (e) Menu

Figure 3: Five representative photos for the dataset

**Caffe** Caffe is a deep learning framework written in C++ [7] and was used for feature extraction and network fine-tuning in combination with its Python wrapper; PyCaffe. Caffe was chosen mainly because of its Model Zoo library, which contains a great deal of pre-trained networks. On top of that, it is a well known and cited machine-vision library that is intended for deep learning vision tasks.

**Sci-kit learn** Sci-kit Learn is a Python machine learning library implementing several state-of-the-art machine learning algorithms. It was used as the main classification toolkit of this research. In order to utilize multi-label neural networks, Sci-kit Learn's developers version (0.18) was necessary.

### 3.3 General approach

Since the images were not labeled, the first approach was to simply assign a restaurant's labels to all its images. Caffe does not support multi-label classification, however, so to quickly explore the potential of this method three experiments were conducted. The first experiment transformed the binary multi-label problem aspect to a multi-class classification problem by creating as many copies of the images as the corresponding restaurant had labels, and assigning one label to every copy. In the second experiment a random label from the corresponding restaurant was assigned to every image. A pre-trained version of CaffeNet, which is a slightly modified version of Alexnet, was used from Caffe's Model Zoo and fine-tuned for both experiments. The fully connected layer classified the images into one of nine classes. The results (F1-scores) were poor. In the third experiment, CaffeNet was used to extract features from the images and Scikit-learn was used to train a Support Vector Machine (SVM) for every label in accordance with the binary relevance transformation method. Even though these results were quite good, the approach of assigning restaurant's labels to their images was abandoned. The main reason for this was due to the underlying assumption that all images reflect every label. When a restaurant has multiple labels, for example `ambience_is_classy` and `outdoor_seating`, not every image conveys information about the label `outdoor_seating` even though that label is assigned to it. An image of a hamburger is not indicative of the presence of an outside terrace. Thus, the model is fed inaccurate information about the images and the dataset becomes noisy.

The second approach was to classify restaurants directly. This approach prevented assigning the restaurant's labels to all of its images and had the advantage of eliminating intermediate

image classification on noisy labels. To examine the potential of this approach, a baseline was established consisting of three parts: image feature extraction, restaurant feature construction and classification..

## 3.4 Baseline

**Image feature extraction** Features were extracted from all images belonging to a restaurant using the first fully connected layer (fc6) of CaffeNet pre-trained on Imagenet. CaffeNet was used because of its relatively simple structure [8] compared to other state-of-the-art CNNs, allowing for fast testing and a reliable baseline. Fc6 contains the most general features of all fully connected layers while still benefiting from smaller feature vectors compared to earlier convolutional layers, resulting in `4096` features per image. Because CaffeNet was pre-trained on ImageNet, more general features are expected to be more representative due to the difference in ILSVRC output classes compared to the YRCC labels.

**Restaurant feature construction** Restaurant feature vectors were constructed by averaging its image feature vectors. This resulted in a `4096x1` vector representation of every restaurant. Average pooling was a simple method of combining features, however it was expected to bear reasonable results due to the expectation that that different images activate different features.

**Classification** Finally, the restaurant feature vectors were classified using Sci-kit learn's SVM in accordance with the binary relevance method, which involved training a separate SVM for every label.

Since results of the baseline were promising and outperformed the architectures of the first approach, the second approach was selected.

## 3.5 Baseline experiments

In order to improve upon the baseline, a number of experiments were conducted regarding optimal network architectures and layers to extract features from, as well as a different method of pooling image features. Finally, a new classification method was examined.

### 3.5.1 Layers

Features were extracted from all fully connected layers to examine the performance differences between them. It was interesting to explore whether more specific features of a network pre-trained on another dataset (ImageNet) would yield better results compared to more general features. In order to try to obtain a richer and more comprehensive representation of restaurants, features from multiple layers were also concatenated into single feature vectors. Finally, a total of 5 different layer configurations were explored: fc6, fc7, fc8, fc6+fc7 and fc6+fc7+fc8.

### 3.5.2 Architectures

Since CaffeNet is one of the simplest state-of-the-art CNNs, deeper architectures were examined as well. VGGNet is substantially deeper (16 or 19 layers, depending on the version) than Caffenet (8 layers) and has a higher single network performance than the comparable GoogLeNet [15]. Because ResNet, currently the latest and best performing CNN, was too computationally intensive, VGGNet became the most interesting network to investigate. Both the 16 and 19

layered architectures were present in Caffe's Model Zoo and used to validate whether the deeper networks performed better. Both networks were pre-trained on ImageNet.

#### 3.5.3 Pooling

As an alternative to averaging the image features to arrive at restaurant features, max pooling was applied by computing the maximum value for every feature. Max pooling was examined because should be able to select the most responsive image for every feature.

#### 3.5.4 Classification

Support Vector Machines have been commonly used for classifying features extracted from CNNs and are often the most powerful classifiers to put on top of a CNN for image recognition tasks [14, 24]. However, since the YRCC was a multi-label problem, training eight different SVM in accordance with the binary relevance method was a naive approach because of relations between labels that were not captured. A Multi-Layer Perceptron classifier was thus examined to improve upon the baseline classification approach by allowing inter-label relations to be captured.

### 3.6 Utilizing CNNs pre-trained on different datasets

After the optimal layer configurations, architectures and feature pooling methods were found that provided the best performing features using a single CNN, focus shifted towards the use of several different datasets to pre-train or fine-tune a CNN on. This was particularly interesting because the images were of different categories, including pictures of scenes and food, as mentioned in section 3.1. CNNs trained on a different sort of data could prove to outperform CNNs pre-trained on Imagenet on certain labels and help to extract more expressive features for different types of images. The first dataset that was examined in addition to Imagenet was MIT's Places [24]. Places was designed for scene recognition and contained 205 different types of scenes and roughly 2.5 million images. Instead of training or fine-tuning a CNN on this dataset, a pre-trained VGGNet16 was selected from Caffe's Model Zoo, thus saving time and guaranteeing a credible training procedure. This CNN was expected to outperform Imagenet CNNs on labels such as `outdoor_seating` and `ambience_is_classy` because of its specialization on scenes.

Considering that food images were a big part of the YRCC dataset, it was sensible to investigate food classifiers as well. There were no pre-trained CNNs intended for food classification in Caffe's Model Zoo and as a consequence a CNN had to be trained from scratch or fine-tuned instead. Therefore, a dataset containing labeled food images was required.

#### 3.6.1 Finetuning VGGNet16 on Food-101

Food-101 [2] was the dataset selected for fine-tuning. It was the biggest labeled food dataset available and consisted of 101.000 food images and 101 food classes. The decision to fine-tune instead of training from scratch was made because 101.000 images was not much data compared to datasets such as ImageNet consisting of 10 million images, and CNNs are data hungry learning algorithms. Additionally, research indicated that fine-tuning provides better results compared to training a CNN from scratch and is able to boost the generality of the features [21]. The *train_val.prototxt* required for fine-tuning in Caffe was not published along with the *deploy.prototxt* and imagenet weights in the Model Zoo, and thus had to be created. This was done by adapting an older VGGNet16 *train_val.prototxt* found on Github [17] and comparing

it to the current *train_val.prototxt* of CaffeNet and the *deploy.prototxt* file of VGGNet16. Figure 4 depicts the learning curve and shows that the network converged after roughly 30.000 iterations. The optimal hyperparameters were found after several experiments and consisted of a base learning rate of 0.001, a step size of 10.000, a gamma of 0.1, a momentum of 0.9 and a weight decay of 0.0005. The first three convolutional layers were frozen because their features were assumed to be general and low-level enough to eliminate the need for additional fine-tuning. A final accuracy of 61% was achieved on the Food-101 test set, which was 5% higher than what the creators of the Food-101 dataset achieved by training an AlexNet from scratch [2].

The Food-101 VGGNet16, Places VGGNet16 and Imagenet VGGNet16 were subsequently used to extract features from the YRCC images, resulting in three VGGNet16 models pre-trained and fine-tuned on three different datasets. This allowed for a fair comparison between the effects of training/fine-tuning on different datasets for the YRCC.

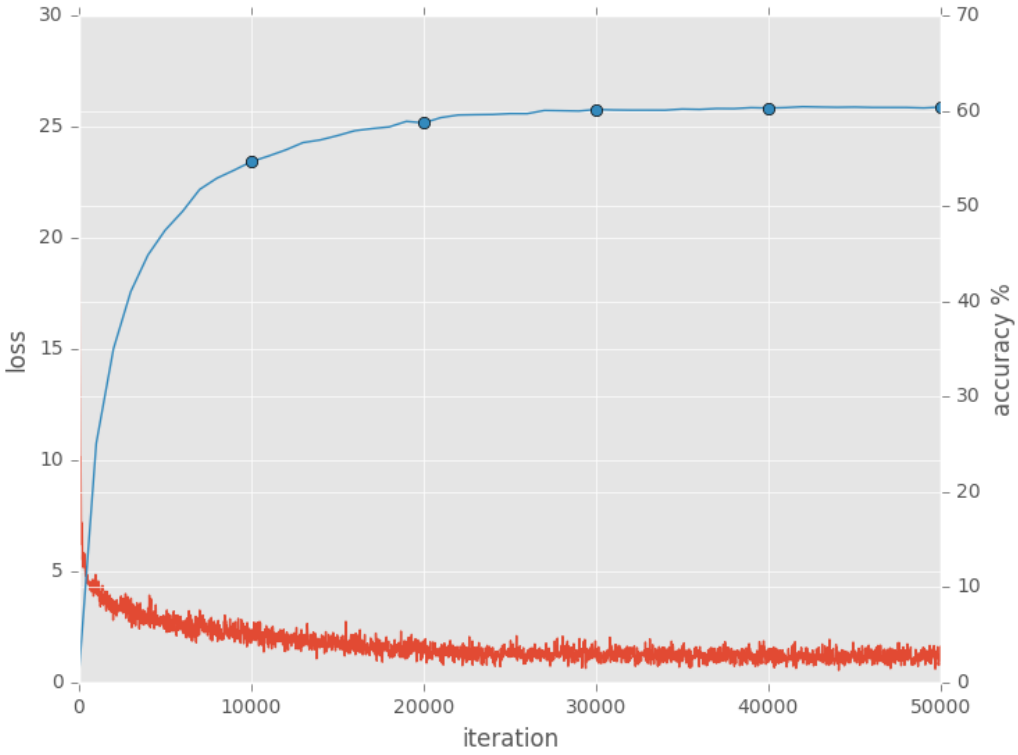



Figure 4: The learning curve of fine-tuning VGGNet16 on Food-101

## 3.7 Multi-Model Ensemble

The final component of this research was to combine the predictions of the individual models in order to arrive at the what was expected to be the best performing architecture that utilized the strengths of all various models. To this end six fusion methods were investigated.

### 3.7.1 Early fusion

The early fusion methods were straightforward and consisted of combining the features in three distinct ways: average pooling, max pooling and concatenation. Late fusion was also performed in three different ways and requires more elaboration.

### 3.7.2 Late fusion

There were three different sets of features, which were extracted by CNNs pre-trained on different datasets, that all had a different neural network classifier trained on them to predict the labels. The first late fusion method consisted of obtaining the decision function for these neural networks per label and taking the average. The decision function was either a positive or a negative number, where positive meant that label was predicted and negative that is was not. The size of the number reflected the certainty of the decision. Thus, if the mean decision function of a label was above zero, it indicated that the ensemble's decision was to predict that label. The second method was based on bagging (see section 2.5) and consisted of a voting procedure where at least two of three models had to predict a certain label in order for the ensemble to predict that label. The third method was based on stacking (see section 2.5 and comprised of training another neural network to combine the decision functions of the individual neural networks. The input to this stacking neural network consisted of features vectors that contained the concatenated decision functions for all labels for the three individual neural networks classifiers. Thus, the size of one feature vector was $1x27$ and was composed of three $1x9$ predicted decision functions.

# 4 Results

As discussed in section 3.1, a great deal of tests were done on the train set before testing on the test set, since it was interesting to see the F1-score per label as well as the F1-micro score (differences are discussed in the following section). Also, extracting features from 1.2 million images was very time comsuming and resource intensive. All tests on the train set were done using 10-fold cross validation in order to utilize the entire dataset while avoiding over-fitting.

## 4.1 Evaluation metrics

The F1-score is an evaluation metric that represents the harmonic mean of precision and recall. Kaggle uses the mean F1-score (F1-samples) to evaluate the predicted labels on the test set by computing the F1-score for every testing example and computing the average over all testing examples. The F1-score per testing example is calculated by using the labels as instances in order to compute the precision and recall. Another method of computing the F1-score is by counting all true and false positives along with the true and false negatives for every testing example and calculate the F1-score (F1-micro) afterwards. Finally, the F1-score per label is used, which is important to determine a model's performance at individual labels. The latter two scores can be computed only over the training set since there are no labels available for the test set.

## 4.2 Results on train set

Table 1 shows the results for the baseline along with two experiments. The first experiment is using max pooling instead of average pooling and does not improve either F1-scores. The second experiment is the use of a neural network classifier and significantly improves both F1-scores. All following results are produced using average pooling to combine image features to restaurant features and a neural network as classifier.

| | Baseline | Baseline / max | Baseline / NN |
|---|---|---|---|
| F1-micro | 0.7693 | 0.7630 | 0.8159 |
| F1-samples | 0.7564 | 0.7452 | 0.7987 |

Table 1: F1-scores of the baseline and the addition of max pooling instead of average pooling and a neural network classifier instead of a one vs rest SVM

Figure 5 shows the results of the experiments regarding different layers to extract features from and reveals no clear improvement over all labels for certain layers. It becomes clear that the models performs significantly better on some labels, for example `has_table_service`and significantly worse on others, for example `good_for_lunch`. Although the F1-samples is rather similar for features extracted from different layers, an increase in F1-micro when multiple layers are concatenated can be observed.

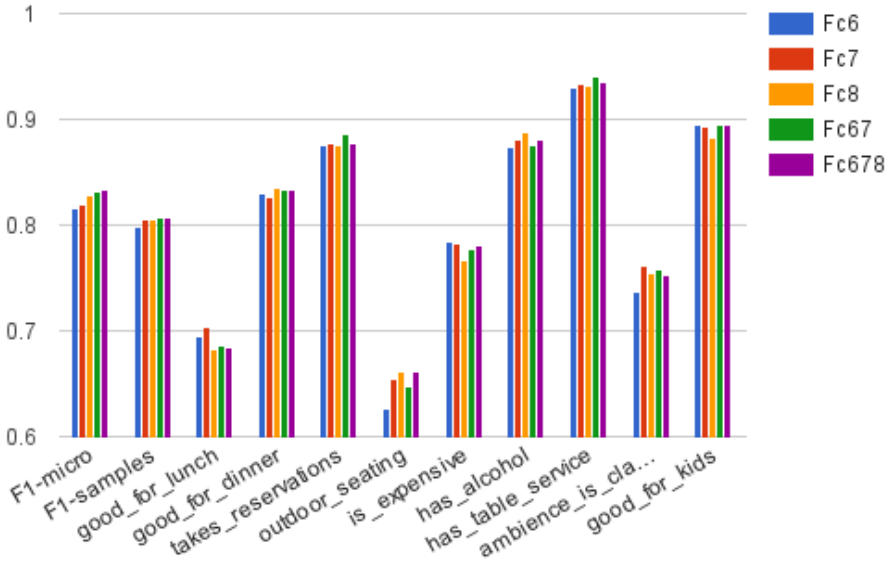



Figure 5: F1-scores on train set using Caffenet trained on Imagenet

The F1-scores obtained after using different CNN architectures can be seen in figure 6 and show that VGGNet16 outperforms Caffenet on F1-micro, F1-samples and all individual labels. VGGNet19 shows no improvement compared to VGGNet16 and has very similar scores.

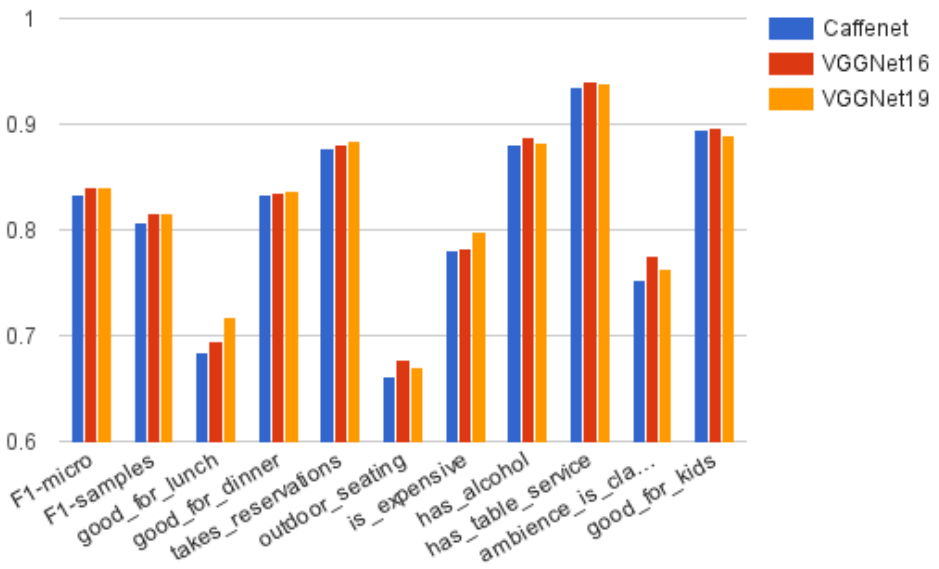



Figure 6: F1-scores on train set comparing Caffenet, VGGNet16 and VGGNet19 trained on Imagenet

Figure 7 shows the performance of VGGNet16 pre-trained on Imagenet and Places as well as fine-tuned on Food101. Food101 consistently obtains a lower score compared to Imagenet and Places on all F1-scores except for the label `good_for_lunch`. Places seems to score the best and outperforms the other models at both F1-scores and several labels.

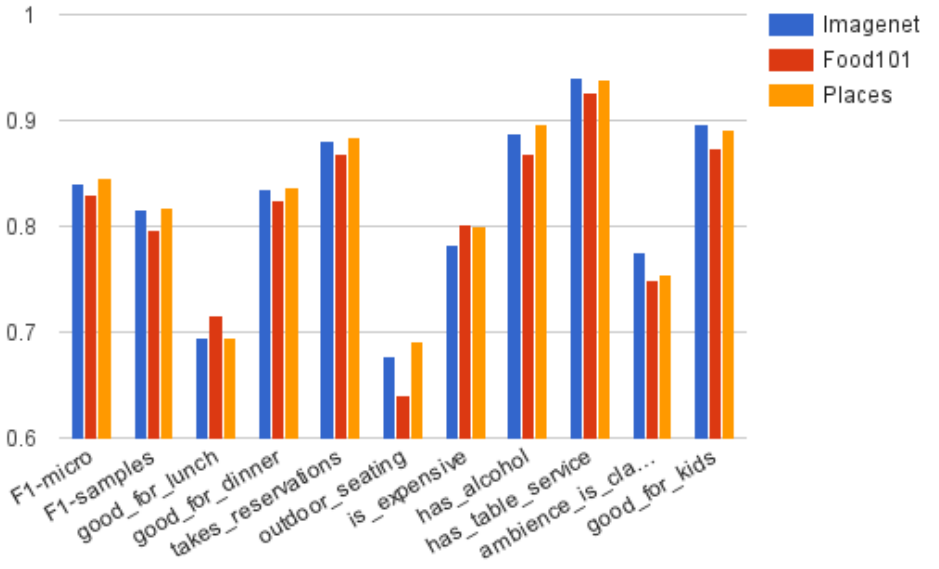



Figure 7: F1-scores on train set comparing VGGNet16 trained/fine-tuned on different data

Results after experimenting with early and late fusion methods are shown in figure 8 and table 2. Average, maximum and concatenation are the early fusion methods, and voting, bagging and stacking are late fusion methods. The multi-model ensemble improves all F1-scores regardless of the fusion method with the exception of average early fusion. The best performing fusion method on the training set appears to be the voting method, where at least two out of three models have to predict a label in order for the ensemble to predict that label as well.

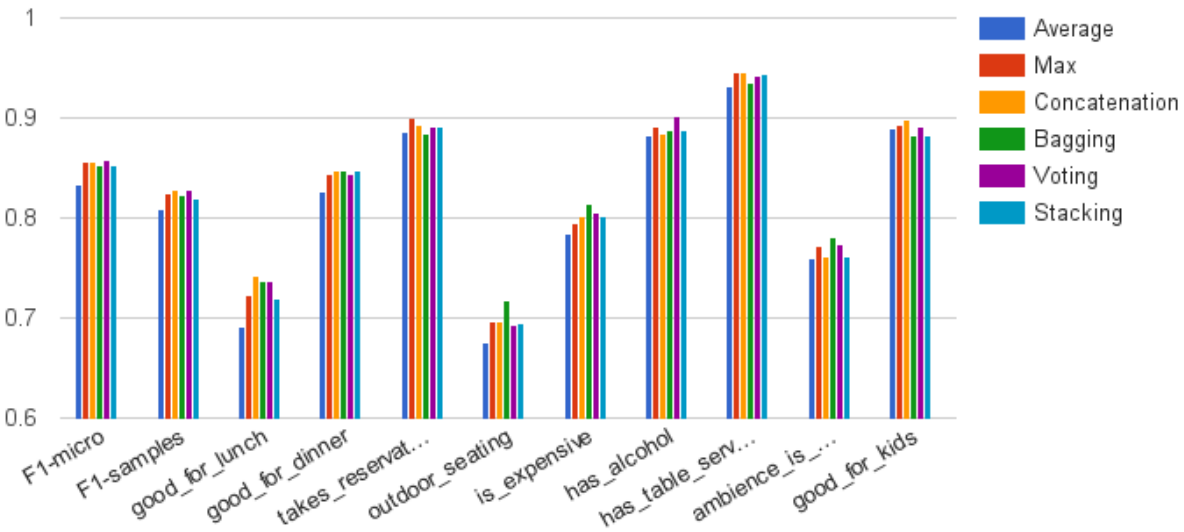



Figure 8: F1-scores on train set comparing different fusion methods

| | Average | Maximum | Concatenation | Voting | Bagging | Stacking |
|---|---|---|---|---|---|---|
| F1-micro | 0.8409 | 0.8559 | 0.8560 | 0.8571 | 0.8526 | 0.8523 |
| F1-samples | 0.8087 | 0.8254 | 0.8275 | 0.8286 | 0.8232 | 0.8192 |

Table 2: F1-scores on train set comparing different fusion methods

### 4.3 Results on test set

Figure 9 and table 3 show the F1-samples score on the test set. A clear improvement in F1-score can be observed as this thesis' approaches progressed. The baseline achieves an F1-score of 0.73 and the best performing multi-model ensemble using bagging or stacking as fusion method achieves an F1-score of 0.82. This is an improvement of roughly 12%. The corresponding rank in the leaderboard to every approach is shown in table 3 and shows that the baseline score is ranked at rank 150 out of the 355 participants. The best scoring ensemble is ranked at place 25, which corresponds to a rise in the leaderboards of 125 ranks.

| | Baseline | Baseline NN | Imagenet | Food101 | Places | Conc | Max | Avr | Vote | Bag | Stack |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1-samples | 0.7317 | 0.7846 | 0.8036 | 0.8011 | 0.8087 | 0.8124 | 0.8119 | 0.8131 | 0.8193 | 0.8204 | 0.8202 |
| Rank | 150 | 130 | 88 | 95 | 80 | 61 | 62 | 56 | 32 | 25 | 25 |

Table 3: F1-samples on the test set along with the ranking on the Kaggle leaderboards for different approaches in chronological order

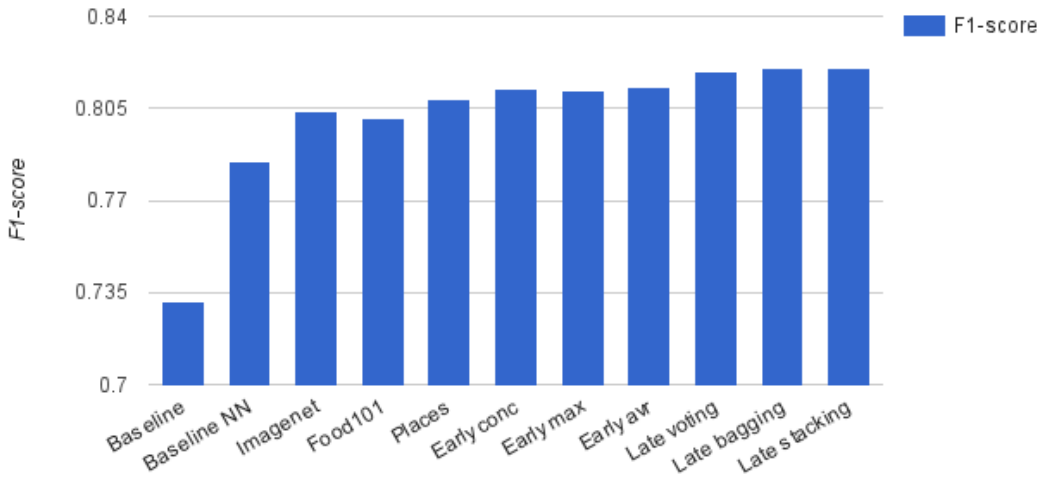



Figure 9: F1-samples on test set comparing different approaches in chronological order

# 5 Evaluation

An interesting observation is the difference in results between F1-micro and F1-samples on the training set. This difference is most probably due to the two metrics calculating a slightly different thing. Since F1-micro counts all true and false positive and negatives, it provides more insight into the macroscopic performance of the model instead of focusing merely on the per-restaurant performance as F1-samples does. This translates into F1-micro capturing for example an overall increase in finding better `good_for_dinner` pictures, whereas that label can not be predicted better per restaurant because there are only nine labels over which to compute the precision and recall. That is, however, captured by F1-samples. F1-micro is also more sensitive to the performance of dominant labels, since they apply more weight by their high number of occurrences which is not the case with F1-samples.

Another surprising result is that VGGNet16 pre-trained on Places outperforms VGGNet16 pre-trained on ImageNet and fine-tuned on Food101 on both the train and test set. An explanation could be that, according to a study by zhou et al. [25], object detectors emerge in VGGNet trained solely on Places. This could enable it to recognize objects, similar to networks trained on ImageNet, as well as learning to recognize scenes causing it to extract relevant features from a broader range of images.

## 5.1 Qualitative analysis

In addition to the quantitative analysis in terms of F1-scores, a qualitative analysis is done as well. The ideal analysis would comprise of ranking the images per label based on the probability that the neural network assigns to them. The top ranked images would thus be the most representable for a label according to the model and could be examined. The problem with this approach, however, is that images are unlabeled and the model is trained on restaurants. Therefore, restaurants were ranked instead of their images and all images of the top three and bottom three restaurants were analyzed for every label. Figure 10, 11 and 12 show two images that were found to be representative for common images in the top three and two images that were found to be representative for common images in the bottom three restaurants for every label. A textual analysis is given as well in the next section.

### 5.1.1 Top and bottom ranked restaurants per label

**Good for dinner** Almost all images in the top three restaurants that the model deems good for dinner depict big, centered white plates with small amounts of food on it. A large number of those plates are not round but rectangle shaped. The photos are mainly taken during the evenings which becomes clear from darker photos and photos made with artificial lighting or candles. Most images from the lowest ranked restaurants are taken during daylight or even from outside the restaurant and commonly depict burgers, counters, menu's other more common types of food. Both the top and bottom restaurants look well predicted.

**Takes reservations** The images in the top three show a great number of white plates as well, just like good for dinner. There seem to be more images of the interior however, which are of relatively fancy looking restaurant with tables that have been set. The bottom ranked restaurants are mainly fast food businesses with cheap looking food that contains wrapping paper, as well as pictures of a counter with people waiting in line. Both the top and bottom restaurants look like a good representation for the label `takes`reservations.

**Outdoor seating** The lowest ranked restaurants for `outdoor_seating`do not seem to have any specific characteristics. The restaurant mainly look more like fast-food kind of restaurants. The top ranked restaurants all have at least one image of an outside seating area. However, a more in-depth analysis reveals that a great number of restaurants labeled with `outdoor_seating`actually do not have a picture that shows such outdoor space. There are also quite some restaurants that do show pictures of outside areas but are not labeled as such, which creates even more confusion. This might be an explanation as to why the F1-score of `outdoor_seating`is so consistently low. If the model trains on restaurants that do not show their outside terrace but do have the label and vice-versa, the model will never be able to correctly learn when a restaurant has outside seating.

**Restaurant is expensive** The images of the top ranked restaurants are almost indistinguishable from the images from restaurants that are labeled as good for lunch. Large white plates, small amounts of fancy food and set tables with white tablecloth. The bottom ranked restaurants mainly sell fast-food like burgers and pizzas and shows images of counters.

**Has alcohol** Every restaurant in the top three has multiple images of glasses containing an alcoholic beverage. Especially cocktails, wine and large glasses with mixed drinks seem to be


(a) Good for dinner


(e) Takes reservations


(i) Outdoor seating

Figure 10: Photos representative of the top three (left column) and bottom three (right column) ranked restaurants per label


(a) Is expensive



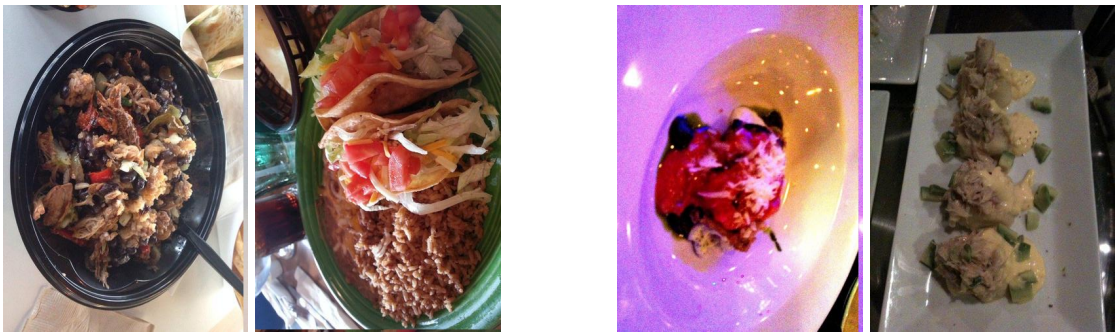
(e) Has alcohol



(i) Has table service

Figure 11: Photos representative of the top three (left column) and bottom three (right column) ranked restaurants per label

(a) Ambiance is classy



(e) Good for kids

(i) Good for lunch

Figure 12: Photos representative of the top three (left column) and bottom three (right column) ranked restaurants per label

important. The restaurants are also quite fancy. The images of the bottom ranked restaurants are mainly taken during daytime and often shows counters, sandwiches and plastic cups.

**Has table service** The top ranked restaurants are very similar to restaurants that take reservations are good for dinner, although they seem to be a lit less fancy. It also looks like there are significantly more photos of people present. The bottom ranked restaurant's images are mainly of fast-food and people waiting in line.

**Ambiance is classy** Photos from the top ranked restaurants are very dark and mostly depict fancy set tables with white or black tablecloth and a lot of white plate and empty glasses of wine. The big white plates are present as well, along with a quite some glasses containing red wine. The bottom ranked restaurants are take-out restaurants with pictures of food in platic containers and food like pizza and hamburgers.

**Good for kids** Restaurants that are labeled as good for kids show a great deal of 'easy foods' with a lot of cheese, meat and sugar. It is also not that fancy with larger portions and less vegetables. It seems like good for kids in this case means food that kids like, not food that is necessarily good for them. The bottom ranked restaurants are very fancy and similar to restaurants that are good for dinner and expensive, which indeed seems like restaurants that are bad to take children to.

**Good for lunch** The top ranked restaurants have a lot of pictures depicting easy, common and fat food. It contains mostly pictures of burgers, sandwiches, pizzas, soups, salads, burritos and wraps. There is often wrapping foil and plastic bags involved and it has a high similarity with images from restaurants that are good for kids. All photos are taken during the day

# 6 Conclusion

This thesis investigated several deep learning architectures in order to predict restaurant categories based on user-uploaded images. Results were obtained on the train set by 10-fold cross validation, allowing for multiple evaluation metrics, and on the test set resulting in one overall F1-samples score. The baseline consisted of features extracted from Caffenet's fc6 combined using average pooling and classified using a one versus rest SVM. It produced an F1-samples score of 0.73 on the test set along with a rank of 150 on the Kaggle leaderboards. The best performing model was a multi-model ensemble that combined predictions of three neural network classifiers trained on features from three VGGNet16s trained on Imagenet, Places and Food101 with a bagging fusion method. This resulted in an overall F1-sample score of 0.82 on the test set and the 25th rank on the leaderboards.

As expected, the use of VGGNet16 improved the quality of the features compared to a simpler and more shallow Caffenet, although there was no additional improvement after using VGGNet19. There were slight F1 increases when extracting features from different layers, especially on the train set when using F1-micro. This indicates that the overall F1-score improves when using features from later in the network and after concatenating them, but the average per-restaurant F1-score does not. The performance difference of VGGNet16 trained on Imagenet, Places and Food101 was lower than expected, although VGGNet16 Places had slightly higher results. However, as hypothesized, the multi-model ensemble did improve F1-score on both the

train and test set. The best fusion method was late fusion, where bagging and stacking produced almost equal results.

# 7 Discussion

Some labels can be not very accurately predicted, for instance `good_for_lunch` and `outdoor_seating`. The cause for this might be that, as is the case with `outdoor_seating`, the data and labels are noisy and relevant patterns can not be learned by the CNN. Although the Food101 classifier shows a small improvement on food-related labels and the Places classifier shows an improvement on the `outdoor_seating`label , the scores for these (and others) are still bad without there being an apparent reason for it.

Another interesting finding is that simply averaging the image feature vectors produces quite positive results. An explanation for this could be that features from frequently occurring images receive a heavier weight in the final restaurant feature vector, unlike max pooling which takes one value from one image per feature. Thus, the restaurant features will be similar to the features of the most occurring images for that restaurant.

Finally, since the three models trained on different datasets produce results that are very similar, especially on the test set, the question arises why a combination of them works so well. This could be explained by the models having a slightly different bias, which is balanced out by combining their predictions. A study by Dietterich investigated why ensembles work better than individual models in general, and found that the use of ensembles also reduce the risk of overfitting due to a larger hypothesis space [5]. One last explanation could be that even though the model's overall performances are similar, they are slightly better at predicting some labels than at predicting others. By combining these predictions, the overall performance on all labels improves.

## 7.1 Future work

An interesting experiment is to cluster the images into five categories using unsupervised machine learning techniques, which would hopefully result in the five clusters as discussed in the Yelp Engineering Blog [19]. Subsequently, different clusters of images could be fed to models trained on different datasets in order to improve inter-model variance. The Food101 CNN, for instance, would extract features only from the cluster of images depicting food, whereas the Places CNN would extract features from pictures of the interior and exterior clusters. The hypothesis is that an ensemble of such models would perform better due to the individual models being better at predicting specific labels within their domain, thus a combination of those models would perform better on all labels.

A straightforward method that should improve the F1-score is to use a more advanced convolutional neural network such a ResNet-152 to extract features from images, since ResNet's top-5 error on the ILSVRC was 3.6% compared to VGGNet's 7.3% top-5 error. Another interesting experiment is to use classifier chains instead of binary relevance and multi-layer neural networks since classifier chains received a lot of attention and proved to be the best performing multi-label classifier compared to common multi-label classification methods in this [10] study. Mult-label neural networks were not compared against, however, so it is unclear whether a classifier chain is able to outperform the multi-label neural network.

# References


[1] Henri Bal et al. “A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term”. In: *Computer* 49.5 (2016), pp. 54–63.

[2] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. “Food-101–mining discriminative components with random forests”. In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 446–461.

[3] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[4] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[5] Thomas G Dietterich. “Ensemble methods in machine learning”. In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[6] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[7] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: *arXiv preprint arXiv:1408.5093* (2014).

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[9] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[10] Weiwei Liu and Ivor Tsang. “On the Optimality of Classifier Chain for Multi-label Classification”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 712–720.

[11] Jinseok Nam et al. “Large-scale Multi-label Text Classification—Revisiting Neural Networks”. In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 437–452.

[12] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Machine learning* 85.3 (2011), pp. 333–359.

[13] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[14] Ali Sharif Razavian et al. “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2014.

[15] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[16] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[17] *trainval prototxt for fine-tuning vggnet16*. 2016. URL: `https://github.com/lfrdm/Masterarbeit/blob/master/train_val.prototxt` (visited on 06/16/2016).

[18] Xiu-Shen Wei, Bin-Bin Gao, and Jianxin Wu. “Deep Spatial Pyramid Ensemble for Cultural Event Recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015, pp. 38–44.

[19] Chuang Wei-Hong. *How We Use Deep Learning to Classify Business Photos at Yelp*. 2015. URL: `http://engineeringblog.yelp.com/2015/10/how-we-use-deep-learning-to-classify-business-photos-at-yelp.html` (visited on 06/16/2016).

[20] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[21] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems*. 2014, pp. 3320–3328.

[22] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *Computer vision–ECCV 2014*. Springer, 2014, pp. 818–833.

[23] Min-Ling Zhang and Zhi-Hua Zhou. "A review on multi-label learning algorithms". In: *Knowledge and Data Engineering, IEEE Transactions on* 26.8 (2014), pp. 1819–1837.

[24] Bolei Zhou et al. "Learning deep features for scene recognition using places database". In: *Advances in neural information processing systems*. 2014, pp. 487–495.

[25] Bolei Zhou et al. "Object detectors emerge in deep scene cnns". In: *arXiv preprint arXiv:1412.6856* (2014).