Project2 - Transfer Learning/Embeddings

Emanuel Buchholz

May 11, 2019

Instructors: Dario Garcia

Important Links
Github
Cats vs Dogs Dataset
Large Movie review Dataset

1 Introduction

For this laboratory report I will get into transfer learning and the use of word embeddings. For the transfer learning part I will use the dogs vs. cats data-set from Kaggle containing 25.000 images of dogs and cats. This data set is from 2013 and back then they estimated the machine learning algorithms of the time could achieve an accuracy of 80% on this classification task. Today I would expect a slightly higher accuracy.
I will use a the VGG16 convolutional layers and make three experiments. The first experiment with random weight initialization of the VGG16 weights and the layers set to trainable. In the second experiment the VGG16 weights will be initialized with weights pre-trained on ImageNet. ImageNet is a large visual database aimed at object recognition research with about 14 million images and about 20.000 categories.
I obtained the pre-trained weights from the standard available weights in the Keras framework. In this second experiment I set the convolutional VGG16 layers to trainable, which means the weights can still be adjusted with back-propagation to the current data set.
Finally for my third experiment I initialized the same model as in the second experiment with the pre-trained weights from ImageNet but I set the convolutional layers to un-trainable. Therefore the network could only learn the weights of the dense layers at the top of the network. This will reduce the training effort significantly, since the layers that can be adjusted is significantly lower.
In general I expect this experiment to work worse than the second one but not significantly worse. The first experiment will likely only converge very late and require a lot of computational power. Therefore we do transfer learning.
For the second part of my report I will look into text embeddings with a large movie review data set provided by Stanford. This data set is intended for binary

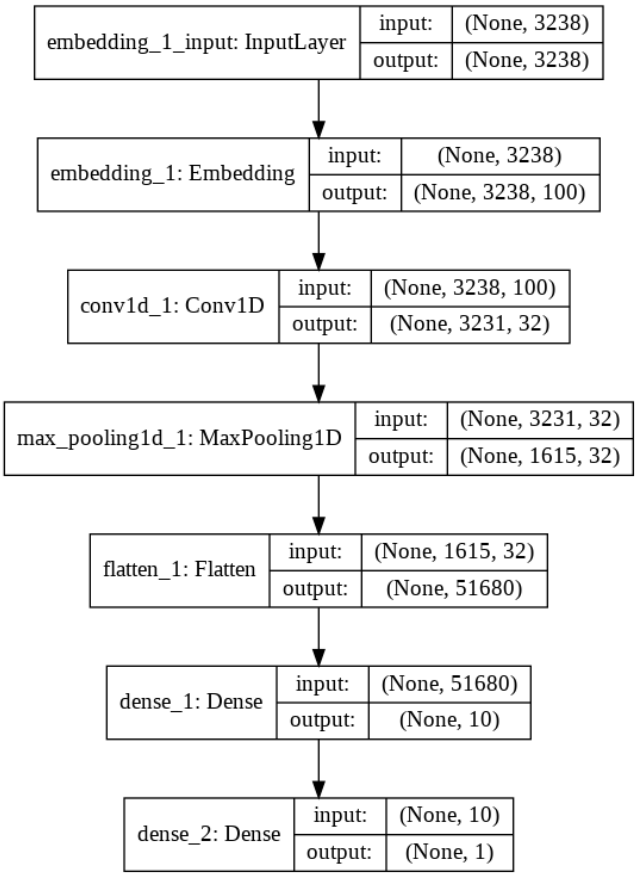


Figure 1: Model architecture of CNN with embedding layer

sentiment analysis (positive or negative). It contains 50.000 reviews pulled from Imdb in 2011 that are labeled with positive or negative sentiment. Additionally it contains another 50.000 reviews that are un-labeled for unsupervised learning. I did not use them though.
The labeled data is already split into test and training data with a 50/50 split. For every movie a maximum of 30 reviews is allowed and the data set also provides a list with links to the review page of every movie. You can find the link to the data set in and more information in Maas et al. (2011).
In the first experiment I created a convolutional network with an embedding layer as can be seen in Figure 1. Then I trained the network without cleaning the text.
Then I implemented text cleaning and extracted a vocabulary from the data set. Then I used this to train an embedding layer on its own using the gensim library.

For the second experiment I used this pre-trained embedding as the embedding layer in my network and then trained the network on the sentiment analysis task.
For my third experiment I used the GloVe word embedding as my pre-trained embedding layer Pennington et al. (2014). Here I used the glove.6B version which contains words from a Wikipedia crawl 2014 and words from Gigawords which is a large collection of words from newswire text data.
Finally I modified the experiments two and tree by allowing the embedding layers to be trainable and not trainable.

2 State of the Art/Methodology

2.1 Transfer learning

Transfer learning is essentially the re-purposing of the learning result from one task to a second task in order to achieve high accuracies and model convergence with significantly lower computing cost. It is very popular in deep learning because deep learning requires a huge amount of resources both computational and in the amount of data. So it is a good way to improve the results if one wants to save computing resources or if there is just not enough data available.

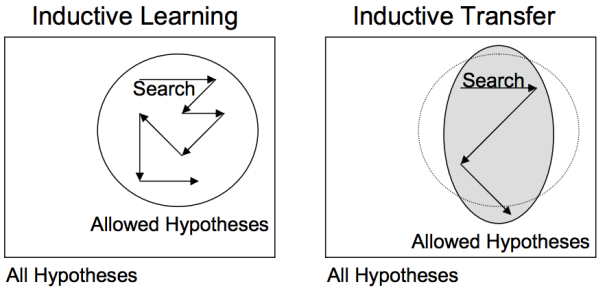


Figure 2: Depiction of Inductive transfer learning from Hassanien (2018)

Transfer learning only works if the features learned from the first task are general and therefore can be generalized to a second model. This is called inductive transfer and means the scope of a model is narrowed on a different but related task. And since the scope of possible models is narrowed the search algorithm can find the optimal solution quicker Hassanien (2018). You can see an illustration of this in Figure 2.
Now for the use of transfer learning in an deep learning application there are two ways to go about it. First the develop your own model approach or the use of a pre-trained model like they are available for many problems.

Develop Model Approach This usually makes sense when the problem is a lack of available data for a specific problem, since you still need to train the

model yourself and therefore expend computational resources.
First you need to select a related predictive modeling problem to your second task, that has an abundance of data available to you and where the input/output data and the learned concepts are similar. Then you develop a model for this large data set that is also similar to the model you will be using to model your second task to which you want to apply the transfer learning. After training you can reuse the model for your second task and fine tune the reused model to the data from your second task.

Pre-trained Model Approach This applies to both, when you want to save on computational resources as well when you do not have that much data.
First you need to look for a pre-trained model that is trained in a way that allows you to reuse it for your second task. For this usually there are many available models from many research institutions as well as maybe you have a similarly trained model available from a previous project. Then depending on your problem you can reuse the whole or part of the model on your second task and then use the data of your second task to fine-tune the model until it performs to your satisfaction.

Summary This second type of transfer learning is common in the field of deep learning and I also used this second type for my transfer learning project. Although my use of transfer learning in the embedding part is similar to the way described in the first part.
In general transfer learning is a shortcut to better model results and we can expect to have a higher start, a higher slope and a higher asymptote (convergence) in the learning accuracy curve.

2.2 Embeddings

For the embeddings part I focused on word embeddings and therefore I will only describe topics relevant to word embeddings. Word embeddings are a very good way to represent humans languages by embedding words in a multidimensional vector space. This allows the transformation of text into a numeric representation of vectors while keeping information about the relationships between words. A big advancement in this field has been achieved by Google researchers in 2013 with the development of word2vec Mikolov et al. (2013). A dimensionality reduced version of the word embedding trained with the movie review data set can be seen in Figure 3a and Figure 3b.

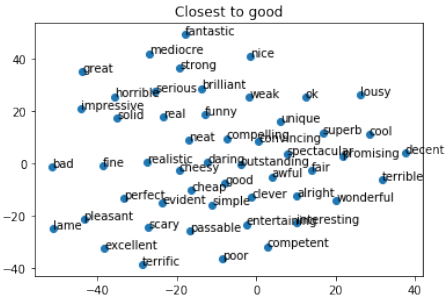
2.3 Text Pre-Processing:

For obtaining a vocabulary of words from a text we need to extract the words from a string of text. There a several things you can and should do, but generally the amount and kind of pre-processing is dependent on your application and data set.

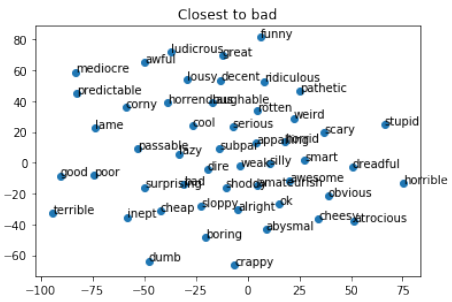
Splitting at white space and removing punctuation: This step is fairly obvious. Since most words are separated by white space, simply splitting at the white space will give you a list of words. Of course there exists the exception of words connected by a '-' or other punctuation. Finally all the punctuation needs to be removed from the words.

Normalizing Case: Finally we can also case normalization to reduce variance in our data even more. This will make the 'The' and 'the' from the beginning and in the middle of a sentence similar. But also meaning might be lost like for example 'apple' and the company 'Apple' are different things.

There are of course many more ways to clean text data, especially with the NLTK package many things are possible. I only used the above methods only, since the cloud does not have outside access and I could not have simply downloaded NLTK packages and also the advantages did not seem to be worth the effort of manually installing them.



(a) Dimensionality reduced sub-sample of the word embedding trained on the movie review data set considering only the 50 closest words to the word good.



(b) Dimensionality reduced sub-sample of the word embedding trained on the movie review data set considering only the 50 closest words to the word bad.

Figure 3: Dimensionality reduced sub-sample of the word embedding trained on the movie review data set considering only the 50 closest words to the word good(left) and bad(right).

3 Experimental Results

3.1 Transfer learning

In Figure 4 you can see the results of the transfer learning with the vgg16 models. As you can see only the model with the pretrained imagenet weights that were set to untrainable showed good performance with, as expected a higher starting

point and a higher end accuracy. I was a bit surprised how bad the model with the preloaded weights that were set to trainable performed. Probably the well defined weights degraded during the learning process, leading the model into a local minima that it could not escape, like the model without any pretrained weights.

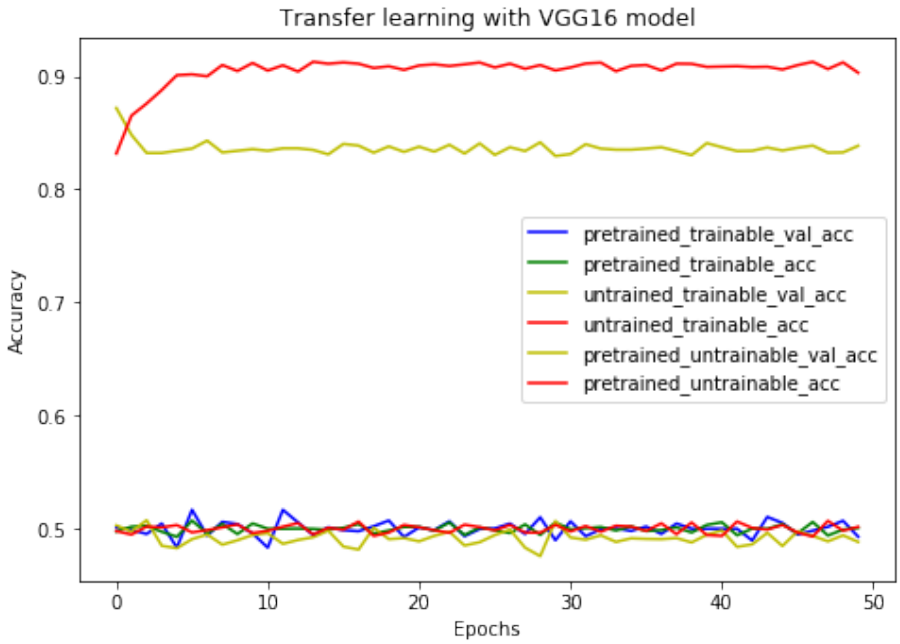


Figure 4: Training results of the transfer learning with the VGG16 model

3.2 Embeddings

In Figure 3a and Figure 3b you can see a sub selection of the 100-dimensional word embedding trained on the movie review data set reduced to a dimensionality of 2. For this I used the t-SNE algorithm developed by Laurens van der Maaten. For the plots I only show the 50 most closest words to the word good and bad. Surprisingly both are in each others closest words. So it appears many reviews are quite vague and use the word good and bad in the same sentence. I had a look into the reviews and these two words are often used in the same review, indicating that the reviewer found parts of the movie good and parts of the movie bad. Essentially this means the hyper-cube we are going to learn with a neural network to separate the two sentiments will have to be quite complicated because there is a large overlap of words with positive and negative connotation in reviews of both positive and negative sentiment towards a movie. As mentioned earlier for the experiments with word embeddings I first did a simple classification task without pretrained embedding or extensive text clean-

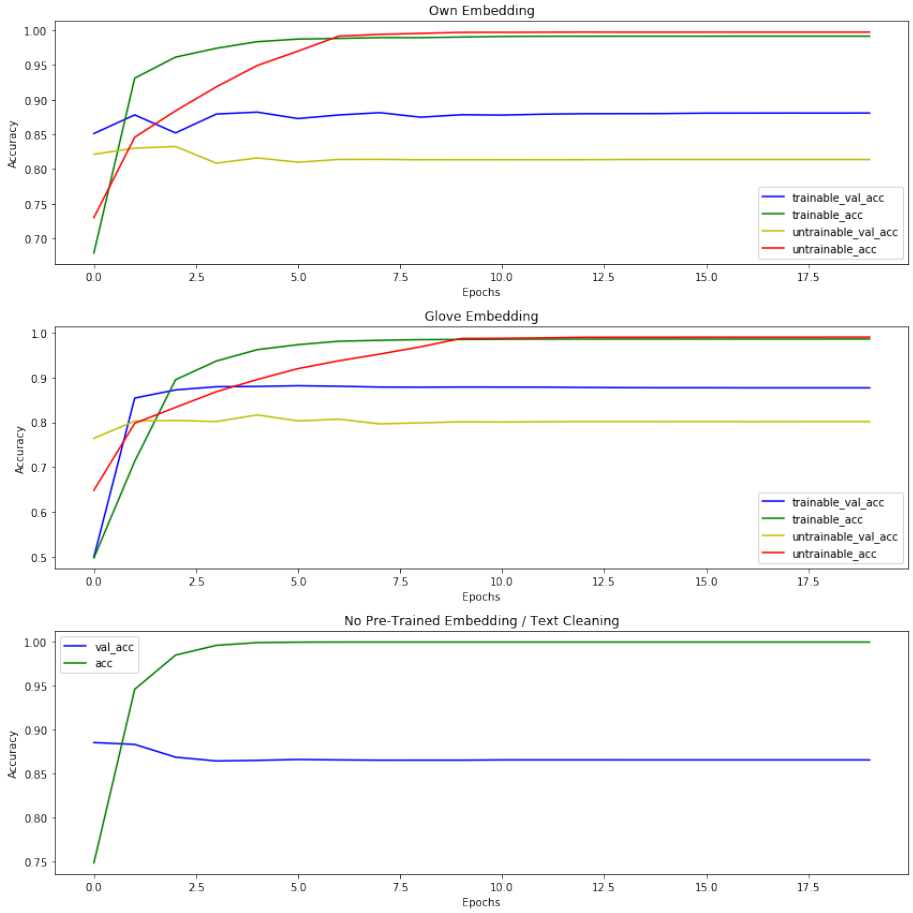


Figure 5: Training results of the experiments with word embeddings

ing. You can see the results of this in Figure 5 in the third subfigure. Here you can see that the network learns the training set well but completely fails on the test set. This makes sense because it does not see the test set and there might be words in the test set that carry important meaning but are omitted because the training set does not contain them. This also explains why the network in subfigure one performs signigicantly better on the test set, because I included the test set in the training of the embedding layer.
Then I trained my own word embedding on the movie review data set and again did the classification task with this embedding once with the embedding layer trainable and once untrainable. I did the same thing using the pretrained glove embedding. You can see the results in sub figure one and two. As you can see both preform quite well and there preformance on the training and test set is very good and quite comparable between both experiments. So I would say no

method has an advantage over the other. What appears to be an important point to note though is, that both of them perform badly if the embedding layer is set to trainable. This is similar as we have seen in the transfer learning section.
So from these experiments we can say that using a pre-trained embedding from the internet brings the same performance as individually training it on the data set. This is probably only true if it was trained on a large enough corpus. Also that we should not allow the embedding layer to be trainable if we pre-trained it. And that a large corpus for the embedding layer is important for good network generalization as we have seen in the tird subfigure.

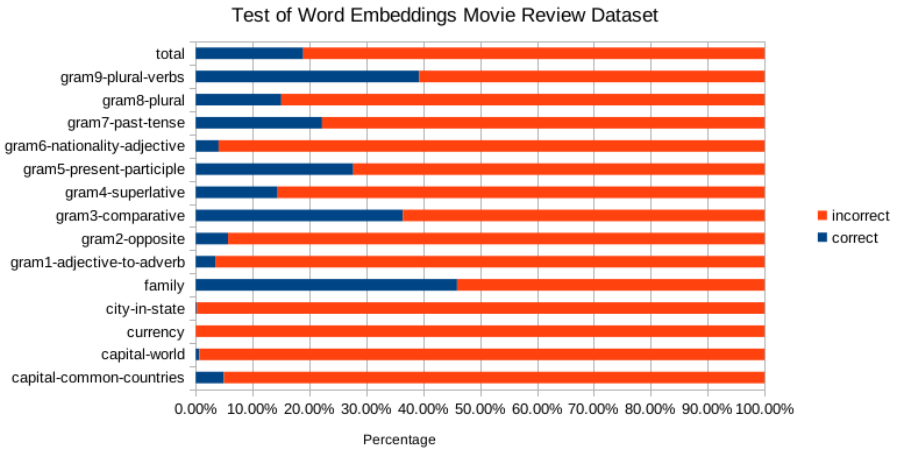


Figure 6: Testing of the word embedding in the matter of King-Men+Woman = Queen with a data set published by google here

Finally I also examined the embedding layer I trained on the movie review data set a bit closer by testing how it encodes the relationships between words. A common example for this is the King-Man+Woman=Queen. For this I downloaded a data set that Google published for evaluating word embeddings and that contains many such relationships. You can see the results of these test in Figure 6. Since learning word embeddings is an unsupervised task, this figure does not say much about the performance of my word embedding, but more about the content of my data set. As you can see it performs nowhere particularly good, but it performs kind of ok in most grammar structures, it performs the best in the family category, probably because movies are often about human relationships and therefore movie reviewers are also talking about it. But it completely fails in the common knowledge about cities and states. Probably because they never come up in the data set.

4 Conclusions

In conclusion I can say that transfer learning works very well and also gives very good results as well for image classification as well as for word embeddings. One of the most important results I found was, that it is very bad for model performance to allow the pretrained layers to be trainable. Maybe slight accuracy improvements can be achieved by making them trainable after the the whole network converged, but they should never be trainable from the beginning. For word embeddings I found that it is not necessary to use an embedding layer trained on the data set one intends to use, a typical data set that can be found online can also be used, as long as it was trained on a large corpus. So finally we can conclude that transfer learning can bring hugh resource savings and accuracy improvements if it is done right.

References

Hassanien, A. E. (2018). *Handbook of Research on Machine Learning Innovations and Trends*.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.