Project Title

Optional Subtitle

Hermanni Hälvä [1]

MSc. Computational Statistics and Machine Learning

Supervisor: Prof. Bradley Love

Submission date: Day Month Year

[1] **Disclaimer:** This report is submitted as part requirement for the MSc CSML degree at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged

Abstract

Summarise your report concisely.

Contents

Chapter 1

Notes

discuss how first trained on small data set

Chapter 2

Intro/Literature Review Notes

Humans are able to extract rich semantic information from visual scenes. For instance, upon viewing a picture of a dog, we may also be able to identify it as a specific breed such as a golden retriever. Further, our understanding of the image benefits from semantic knowledge that is not captured explicitly in the visual features of a specific image. For example, we also know that the dog belongs in the category of mammals which, in turn, are animals and thus living entities. While this type of hierarchical semantic visual understanding comes to us effortlessly, it remains a challenging task for computer vision. In fact, most image classification models are trained on data sets with single, mutually exlusive labels and thus the learnt feature representations do not account, for example, for both cat and dog being four-legged animals. Why is this a problem? Generalization? An exception to this is the area of knowledge transfer and related tasks such as zero-shot learning in which the aim is to predict labels of previously unseen classes of images; a popular approach for zero-shot learning is to borrow strength from, say text data, to create a semantic space that embed all the possible labels of images including those not seen previously. Mapping between images and the semantic space is then learned using the 'seen' images. Once this mapping is learnt, it can be used to transform previously unseen images into the semantic space and then apply some distance measure to label it as the category thats cloest in the embedding space. We postulate that the same idea could also be employed in the simple image classification tasks to achieve better generalization performance. In particular, we will train a deep learning model for image classification against a hierarchical semantic embedding derived from the WordNet database which, as we will show, will correspond to regularizing the model weights to account for these semantic relationships. We will use the recently developed poincare embeddings as the resulting embedding space can capture both the similarity between the possible labels as well as the hierarchical semantic relationships encoded by WordNet graphical model. The learnt model is then transferred to perform standard image classification by adding a final softmax classification layer and fine tuning this agains the ImageNet database. Our results show . . . hopefully some improvements over training the model only against ImageNet which illustrates the benefit of incorporatig semantic knowledge..

PAPERS ON PSYCHOLOGY OF VISUAL PERCEPTION & SEMANTICS

PAPERS ON SEMANTIC KNOWLEDGE TRANSFER & ZERO-SHOT LEARNING

These zero-shot learning papers are very much related to ours in the sense that they try to achieve generalization via word embeddings into unseen image classes. The DeViSE paper is probably the most similar though others relevant too. In fact, some of them already used this idea before the deep learning revolution.

Large Scale Image Annotation: Learning to Rank with Joint Word-image Embeddings [**?**]

Overview

This is a pre-deep learning paper, on how to handle efficiently annotation of 'new' large scale data-sets such as ImageNet. Approach is to learn to represent images and annotations jontly in a low dimensional embedding space, idea being that low-dimension at test time translated into faster model. Indeed, a pre-deep learning methods such as performing knn in image feature space suffer from the curse of dimensionality. To do this they use a loss function that allow the model to learn-to-rank i.e. to optimize precision of k-top annotations. In practice, their model performs linear mapping into joint embedding space (i.e. annotations and images are multiplied by their own matrices of equal dimensions).

Thoughts:

- while impressive at the time, no longer state-of-the-art, but their point about curse of dimensionality may still be important even for DL. Say if a DL model has a softmax of too high N, it becomes harder and harder to separate out different classes so word embedding may help here
- The model used by the authors is just a linear matrix transformation. Thus DeViSE paper can be seen as a big improvment over this.

Zero-shot Learning Through Cross-modal Transfer [?]

Overview

Motivated by the idea that humans have the ability to identify unseen objects even if they know about that object from having read about it. The authords introduce a model that can predict both seen and unseen classes: 'without having ever seen a cat the model can say whether it is indeed a cat or another category it has seen during training'. Large unsupervised text data is used to create word embeddings and then images are mapped into this space using neural networks. Prediction of unknown classes works by determining whether the test image is on the same manifold as known examples - this is based on outlier detection. One advantage over previous works is that unsupervised text corpus can be used rather than a manually constructed word embedding space.

The model used to create unsupervised word embedding is based on [?] (see above), which can be seen as earlier version of GloVe. They perform this on Wikipedia text data to create word embeddings that capture local and global context. Image features are then computed in unsupervised fashion using orthogonal matching pursuit. With the word and image feature vectors available, a two layer neural network is trained on the images with known classes into the word embedding space. The authors then use t-sne to visually illustrate that if unseen classes are fed into the model, the predicted word vectors will not be close to the vectors of known classes (which are clustered in turn). The closest known classes to the zero-shot classes however give idea of their semantics. In practice, to accomplish above we need to first detect whether an image is of a known class or of unknown class, and for this purpose a binary novelty random variable is employed (this is becuase otherwise would never predict any of the unseen classes if only used training data of seen classes). An outlier detection (based on Gaussian distribution) to predict whether an image is of unseen class given its predicted word embedding vector. If image is predicted as seen, a softmax classifier is used to predict its label, whereas, if image is predicted as new, an isometric Gaussian is assumed around its word vector and its class is assigned based on likelihood i.e. distance betwen predicted word embedding and full-word embedding space that contains also the unseen classes (this is the classic zero-shot trick using word data). The equations in the paper make this much clearer so will add those. In training, CIFAR-10 image data is used with 2 classes omitted for testing. The results depend strongly on the threshold used to determine whether image is in an unseen or seen class.

Other interesting points: The authors show how the zero-shot learning can be framed in a fully bayesian way. They also compare the novelty detection in word embedding space as above, to doing novelty detection in image feature space and find the former superior as it adds the additional information from semantics.

Thoughts:

- the zero-shot approach here may not be fully relevant to us but I still wrote it up so have a general idea of why word embeddings are used in these papers. Further, I do find this an interesting application and if we have time, may want to consider zero-shot learning

- no deep models used here really though they do call their two layer neural network a deep model, different times . . . no convolutions though

- hierarchical information is not used

- a major problem is that the threshold used to discriminate between known and unknown class creates an inherent trade-off between the ability to predict these two. The DeViSE model improves on this (see below)

- a lot of useful references to go through from this paper, and some other intresting ideas not covered above may be worth exploring later

DeViSE: A Deep Visual-Semantic Embedding Model - Frome et al. 2013

Overview

Typical object classification models treat all the categories unrelated, via N-way softmax. This leads to models that 'cannot transfer semantic information about learned labels to unseen words or phrases. Solution this paper proposes is to use both standard image data and then an unrelated large unannotated text data to learn semantic information. In particular, the model maps image inputs into this rich semantic embedding space. The results show similar performance to standard state-of-the-art DL models, but with a significant improvement in that much less 'semantically unreasonable' mistakes are made. Perhaps more importantly, they show this joint training allows the model to generalize to 20000 visual categories, despite having trained the model on just 1000 categories.

DeViSE extends work of Weston et al. (above) as it allows non-linearities and also capture semantics from text that's not contained in the image lables, hence allowing for zero-shot generalization. It also improves on the Socher et al. paper by using a deep model and avoid the trade-off which that paper has in predicting seen and unseen classes. Socher et al. also combine several different models whilst this one uses a unified model that only uses embeddings. Several other previous papers have used WordNet to build semantic representations; they authors here use a large unannotated text data which they claim is superior.

The modelling approach begins by first training the skip-gram model which predicts the adjacent terms in the text for each word and then creates an embedding on this. This model was trained on 5.4 billion word Wikipedia data set. Image-labels were then mapped into this vector representation. Next, a DNN was used to map images into this embedding space by removing the final softmax layer and instead using a similarity metric. More precisely, a combination of dot-product similarity and hinge rank loss was used (following Weston et al.); this was found to be better than L2 loss. During model testing, image is projected into the embedding space and the nearest label is found using a hashing technique. The corresponding image-net synset is then found for this embedding.

The model is then used to perform zero-shot learning and compared against few baselines: state-of-the-art DL model and the authors' DeViSE model but using random embeddings rather than learnt word embeddings. The authors claim better results than standard DNN on standard classification task on imagenet, though I am really not convinced by this as the differences are tiny and doubt statistically significant. However, on zero-shot learning, DeViSE does really seem superior.

Thoughts:

- quite similar to our approach as we will also look to map image labels into word embedding space
- we will also follow this by removing the softmax layer from our choice of DNN and instead use some similarity metric
- we need to thus think what will be our final loss we will use, probably try several different
- need to think how to perform testing since predictions just give embedding location. Not sure if the hashing technique is the best way to do this.
- I really dont see why the use of the wikipedia data is superior to wordnet. Since wordnet seems to already correspond to how humans build a taxonomy of visual objects (see earlier 'psychology papers') then surely that would be the preferred approach? In particular, the wikipedia text is clearly larger but will have a lot of noise, and more importantly it may not be predictive of the relationship of how visual features correlate. For instance, the words 'bird' and 'sky' are likely to end up near each other based on the wikipedia data, but the two dont visually

resemble each other so perhaps it's not good for them to be close if the aim is to influence what visual features are to be learnt.

Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs - Wang et al. 2018

Overview In previous literature, zero/few-shot learning has been achieved via knowledge transfer. Two different routes have been used for knowledge transfer. The first is to use implicit knowledge representations in the form of semantic embeddings create from some adjacent text data. Accoding to the authors, the generalization power of semantic models is limited, partly by the mapping models themselves. Further, there is no easy way to learn semantic embeddings from structured information such as knowledge graphs. Indeed, the second approach to zero-shot learning has been to use explicit knowledge transfer of rules and relationships. A simple example given is to learn a separate classifier for different compositional categories of a visual object.

This paper's novel contribution is to use both implicit knowledge representation (i.e. word embeddings) and explicit ones (i.e. knowledge graph) to learn a visual classifier. This is done by constructing a knowledge graph where the nodes of the knowledge graphs are the semantic word embedding representations, and are connected to each other by by edges that represent the relationships between the words. The node semantic embeddings are created using GloVe. Graph Convolution is used to pass messages in the graph between the categories according to the knowledge graphs, which in one of the experiments is just the WordNet sub-graph. Essentially this approach generates a new deep logistic classifier for each object. The visual features are extracted from inception V1/Resnet50 model, depending on the experiment. The results of the paper show very large improvements over DeVISE and other state-of-the-art in zero-shot classification.

Thoughts:

- the authors say its hard to learn semantic embeddings from structured information but actually Poincare embedding should allow this since they precisely try to represent hierarchical data in embedding e.g. wordnet.

- what exactly is wordnet SUB-graph? need to check
- the performance of this model is very impressive; I wonder if it we should also attempt how our model does in zero-shot learning

PAPERS USING WORD HIERARCHIES IN IMAGE CLASSIFICATION

Learning and Using Taxonomies For Fast Visual Categorisation - Griffin and Perona 2008

Overview

Pre DL-era paper in which the authors attempt to move from linear computational cost in number of categories of one-vs-other classifier to logarithmic cost by considering a tree-like hierarchical classification. This is particularly importan for large number of labels, say 10^4. The authors conjecture that humans are able to perform fast visual categorization by moving down a hierarchical taxonomy an hence avoid considering irrelevant classes e.g. to classify an 'object' as a dog first decide if something is an animal or not and after that if the animal is a dog or a cat. To do this need to create hierarchy of categories as a binary tree. This is done using Spatial Pyramid Matchin on the images and their labels.

Thoughts:

- hierarchy here is in the classification model though the idea is similar to ours that humans hold visual categories
- the model here is very complex in that it requires several pre-processing steps e.g with SIFT so very different from DL approach
- rather than using wordnet hierarchy they calculate hierarchy from images and their labels. Using wordnet embedding would gives us the ability to draw strength from categories not necessarily in our images (though if we use imagenet, they should mostly be there?)

NLP WORD EMBEDDING PAPERS

These papers provide important background on word embeddings. Main idea is to represent words in vector space such that distance between any two vectors reflects the similarity of those two words (semantically).

Improving Word Representations via Global Context and Multiple Word Prototypes - Huang e

Overview

Main point is that in order to cluster words vectors appropriately, need to consider both syntax and semantics. Most previous approaches have only done this in the context of the local problem, but employing a global data can provide more accurate resuls, as it would better capture semantics whilst still contolling also for syntax. A joint model is hence used. Indeed, the authors show that this joint model provides better correlation with human similarity judgement of words.

More specifically, 'the model jointly learns word representation while learning to discriminate the next word, given a short word sequence (local contexti, syntactic information) and a document of text the sequence appears in (global context, semantic information)'. Two standard neural networks are used to predict scores that reflect the likelihood of the most recent word occuring given the local and global context. Sum of the two scores creates a total score for the most recent word, which is then used to construct a hinge-loss type of function to discriminate against any other possible word that could have occured, which is then minimized. *Thoughts:*

- This seems like a pre-cursor to GloVe

MULTI-LABEL CLASSIFICATION PAPERS

According to current plan, we are concerned with typical single-label classification. Nevertheless, multi-label image classification is relevant to us since those papers have to explicitly try to learn relationships between many related categories in objects. Different papers below approach this in different ways and have somewhat differing aims. They do share our goal of trying enforce some way of semantic understanding into DL papers. I do sometimes feel that multi-label classification is 'under rated'; it's much closer to how I imagine that humans process visual data.

YOLO 9000 - Redmon and Farhadi 2016

Overview

This is a object detection paper so in a way quite different from ours i.e. it tries to predict locations of all objects in the images and then label those. This paper is relevant to us due to its use of hierarchical labelling, which the authors use to combine distinct datasets of unequal hierarchy into one dataset, so their purpose of hierarchies is though very different from ours. More specifically, in this paper the authors combine imaages from classification data sets and object detection (includes localization and possible multiple objects). Problem is that imagenet has very specific labels (c.f. subordinate in above paper) whilst detection data set has only very generic labels e.g. dog (c.f. basic labels above paper). Need a coherent way to marge these different level labels, to create the joint data set. Also, say 'norfolk terrier' and 'dog' may not be mutually exclusive in photos thus cant merge using just a single softmax like in DL typically. Thus the authors instead create a hierarchical tree of the imagenet labels (which are based on WordNet). To classify with their hierarchical tree, predict conditional probabilities at each node i.e. probability of each hyponym of that synset given that synset. e.g. P(norfolk terrier—terrier), P(Yorkshire terrier— terrier). Can then calculate marginal probabilities by traversing upwards through the graph. This allows the authors to combine COCO and ImageNet data sets and efficiently do detection on over 9000 labels with only marginal performance drop. The authors also show that with the hierarchical model, the performance of the model degrades gracefully when it sees for instance a dog but isnt certain about the breed of the dog

Thoughts:

- the idea of hierarchies is more similar to us here than in the Peterson paper
- would be interesting thus compare our models performance to this one, though it may be hard given the weird training done here
- the hierarchical approach is more of a 'means' here and in itself is not much analysed. e.g. there is no discussion of reprsentations
- Whilst how theyve built the tree from wordnet is impressive, it only captures hierarchies whilst a word emebedding space may be able to capture more complex relationships
- I find their point about graceful degradation fascinating and will look to explore that as well. Very curious if a hierarchically trained version of our model turns out to be more robust to adversarial attacks.

CNN-RNN: A Unified Framework for Multi-label Image Classification - Wang et al.

Overview

In reality, visual views rarely ever contain just a single object with a unique label; rather, we perceive rich semantic information in even the simplest views. Single label classification fails to capture this, hence multi-label classification. Usual approach to multi-label classification treats it as multiple single label classification problems. This fails to capture the dependency between multiple labels e.g. sky and clouds usually appear together but cars and water shouldnt. Machine vision has in the past captured these type of dependencies using markov random fields but such grid models only really control for pairwise dependencies. Cannot handle complex higher order relationships in images. To do this the authors of this paper use RNN on multi-label data and show that this significantly improves classification accuracy.

Usually CNNs share features across different classifications, but the problem here is that small objects in images are hard to classify because the features are built on classifying the whole image in the best way possible. RNN helps in this because, as the paper shows, it implicitly creates an 'attention' model where the classifier focuses on different areas of the image based on the RNN memory state. More specifically, this is done by learning join image-label embedding to model semantic relevance, where the image embedding is the lower layer of the CNN. These are projected into same sub-space as label embeddings, and the LSTM memory thus captures higher order dependecies in this embedding space in particular. Model is best understood from figure 4. In prediction stage, beam search algorithm is used because markov property is not satisfied. And to be clear, training is done on multi-label data sets such as MS-COCO.

Results: state-of-the-art on multi-label since can weed out labels that can't possibly co-occur. On 1000 label data set the performance is poor because the DL model is trained on image net, which doesnt have concepts such as actor/actress that occur in the large multi-label data-set On MS-COCO data set poor performance with few tiny objects such as toaster/hair dryer that have little dependence on other categories. The authors also show nearest neighbours in label-image joint space to illustrate that the model indeed captures semantic similarity. They also show that (using de-convolution), there is implicit attentional mechanism where first the model focuses on entire image and then moves on to smaller parts.

Thoughts:

- A big difference between all these multi-label approaches and ours would be that here semantics are really controlled only by what's visible in images, whilst we could use entire corpus of text / ImageNet hierarchy even for a single image.

- On the other hand, our model wouldn't be able to do multi-label classification as it's hierarchical 'understanding' would derive entirely from word embeddings..

- A little like that attention here, I wonder if we could also visualize if using hierarchical model changes the regions of the image that the model focuses on vs. non-hierarchical model

- the observation about toaster/hair dryer being difficult to capture semantically is interesting. I wonder if our word embedding approach would suffer from similar?

- bit like the nearest neighbour label prediction here, we should attempt also to predict into the embedding space after training the model. I really wonder if we could thus train on single label data but actually somehow accomplish multi-label predictions (based on predicted embedding space). I guess this would be kind of what the zero-shot learning papers (see the section on those papers) attempt to do using word embeddings.

Other relevant papers and intro stuff

Computer vision researchers have already long before the advent of deep learning, been interested in multi-label classification as it would greatly enhanc our ability, for example to search and reterieve large quantities of image data [**?**]. Further, multilabel classification much closer corresponds to how humans perceive complex visual scenes and thus developing models with rich semantic ability would be a valuable step for development of artificial intelligence technology such as autonomous vehicles. Multi-label classification already of interest before deep learning: [**?**].

Similarly, word embeddings have been used extensively to capture image semantics much before deep learning. As an example [**?**] perform unsupervised image auto-annotation with a probabilistic model in which a joint latent space is used to represent co-occurence of image features and words. More specifically, the authors constraint the latent space to be mainly defined by word-features as they argue that the semantic relationships between words is richer than that of image features, and second, co-occurence of words in text is semantically more meaningful than that of visual features. The authors' first point may be less relevant for the state-of-the-art today as convolutional neural networks have shown the ability to build complex fvisual features. Nevertheless, if we train model that captures full semantic relationships in large sample text, this should be richer still; for instance, ImageNet is represented by a Directed Graph rather than just a tree hierarchy. The second point too remains apt, image feature co-occurence indeed may not mean much (though deeper layer ones do probably more so in CNNs). The presence of varying background and other objects, makes it difficult to maintain semantic similarity on the basis of visual features alone.

How is multi-label classification different from semantic segmentation / object detection? Not trying to locate/bound any object but rather concerned with labeling. Why would we need something other than YOLO9000 (Redmon and Farhadi 2016). Further, in our approach we are not trying to perform multi-label classification, rather we still perform single label classification but trying to benefit from semantic dependencies in training…

Is it really a reasonable assumption that semantically similar words should be close to each other in image space. Perhaps true for e.g. cat and a dog. But what about sheep, clouds or snakes and cables. Should they be close to each other in an optimal embedding space?

In Bayesian framework, the use of embedding would probably corresond to some prior that is the wordnet graph

Chapter 3

Implementation Ideas

- which loss function to compare predicted and ground-truth embedding vectors. Dot product hinge loss as in DeVisE?
- at test time, how to predict labels i.e. to find nearest neighbour? Some tree or hashing e.g. DeViSE
- which baselines to use? Standard inception-v3, random embedding, word2vec embedding,
- what evaluations to make? resonableness of errors

Chapter 4

Artificial Neural Networks & Deep Learning in Computer Vision

Even though deep learning is often viewed as a new technique, in reality the recent breakthroughs are underpinned by decades, if not centuries of related research. For instance, earliest artificial neural networks, such as Rosenblatt's Perceptron [?] from the 1950s, are closely related to linear regressions dating back to Gauss [?]; these models are all of the form $f(\mathbf{x}, \mathbf{w}) = \mathbf{x}^t\mathbf{w}$ where $\mathbf{x}$ is a vector describing some input covariate and $\mathbf{w}$ model weights. Much like in linear regression, the aim is to learn a mapping $f(\mathbf{w}, \mathbf{x}) = y$ that defines the relationship between the input $\mathbf{x}$ and some category y it belongs to. In a simple problem, y would be binary and the weights $\mathbf{w}$ would be learnt such that the resulting hyperplane could linearly separate the data into the two classes based on the input features. These early models were largely restricted by their assuming linear separability and there was also no computationally feasible way of training the models at the time [?], [?] but they form the basis for nearly all Artificial Neural Networks (ANNs).

Several steps were taken to increase to complexity of these models and to allow for non-linearities, many of them initially inspired to some degree by biological neurons [?]. For instance, real neurons typically only fire once action potential exceeds a specific threshold [?]. In ANNs, a reminiscent behaviour is attained via activation functions on top of a neuron outputs, most typically in the form of Rectified Linear Units (ReLU), which apply the following non-linearity: $f(\mathbf{x}, \mathbf{w}) = max\{0, \mathbf{x}^t\mathbf{w}\}$. ReLUs also improve the representational capacity of a network by imposing sparsity which can make it easier to disentangle the data [?] and hence lead to faster convergence of training [?].

Another insights borrowed from neuroscience was that intelligence stems from groups of neurons acting together rather than from the behaviour of individual neurons [?]; this idea is behind deep ANNs where several layers of neurons are connected to each other and numerous neurons are present in each layer. Below equations and Figure 4.1 give a simplified example of an ANN with two hidden layers:

$$\mathbf{H_1} = max\{0, \mathbf{WX} + \mathbf{B}\} \tag{4.1}$$

$$\mathbf{H_2} = max\{0, \mathbf{VH_1} + \mathbf{C}\} \tag{4.2}$$

$$\mathbf{F} = \mathbf{ZH_2} + \mathbf{D} \tag{4.3}$$

where $\mathbf{X}$ is an $D \times N$ input data matrix that holds the N different observations in columns and D is the dimension of a single observation, which for image data is often a flattened array of the image pixels. This matrix representation of input allows several datapoints to be fed through the network simultaneously, which is what is done in practice. $\mathbf{W}$, $\mathbf{V}$ and $\mathbf{Z}$ are the weights matrices of the two hidden layers and output layer respectively. The number of rows in each of these matrices is the number of neurons in that layer, whilst the number of columns corresponds to dimensionality of output from the previous layer. Notice also that constant bias matrices $\mathbf{B}$,

C, and **D** are added to the neuron outputs. These bias matrices are akin to intercepts in regression analysis and increase the representation ability of the model. Usually all the columns of the matrices are identical such that the same bias values are added to each input observation. We can see that mathematically these matrix operations are just affine transformations on the input, followed by ReLUs, which are applied elementwise. The resulting output of each neuron is thus a matrix, here **H1** and **H2**, where each row corresponds to an output from a specific neuron, calculated separately in the columns for each input vector. Notice also that ReLUs are not added on the output, rather the output layer transforms the representations of the hidden layer into a desired shape of output. For instance, for binary classification **F** would be of $2 \times N$, and the two output values for each input would reflect the relative likelihood of the two labels.

The ANN model we have described thus far is known as Feedforward Network or a Fully Connected Network, which refers to all neurons being connected to all other neurons in the preceding and succeeding layers. This is an important point as it fascilitates a hierarchical structure in which neurons in the later layers may combine features from earlier layers to create more complex features in turn. Relatedly, this architecture enables distributed representation in which several types features can be combined in different ways to represent an exponential number of different inputs [**?**]. As an example, if we had squares, rectangles and circles and each of them could be either red, green or blue, then we have 9 possible visual objects, yet all the possibilities could efficiently be represented by combining three color neurons with three shape neurons [**?**]. **see goodfellow ch 15**.

In a typical classification problem we have n possible labels for each input and wish to predict a probability distribution over them. In these applications the outputs of ANN are transformed into 0 to 1 range. More formally, consider that the output from above ANN for a single input $\mathbf{x}$ is the $n \times 1$ vector $\mathbf{f}$; we then require, that each element of $\mathbf{f}$ is between 0 and 1 and that

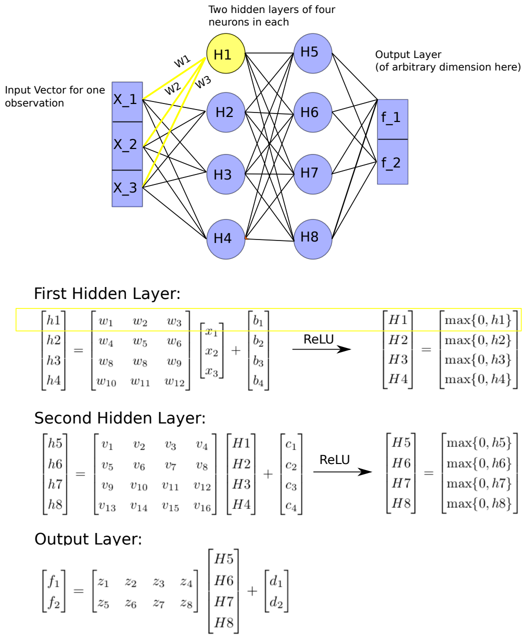
Two hidden layers of four neurons in each
Input Vector for one observation
Output Layer (of arbitrary dimension here)
X_1
X_2
X_3
W1
W2
W3
H1
H2
H3
H4
H5
H6
H7
H8
f_1
f_2
First Hidden Layer:
ReLU
Second Hidden Layer:
ReLU
Output Layer:

Figure 4.1: A simple Feed Forward ANN with two hidden layers. This figure illustrates a graphical model for Equation 4.1 - 3.3 as well as explicitly showing the matrix operations performed by the different neurons on a single vector input. The output dimension is set arbitraily to be a 2 × 1 vector, which could for instance be used as class scores in binary classification; in practice, the dimension will be problem dependent.

$\sum_1^n \mathrm{f}_i = 1$. The most common way of doing this is to assume that the output of the ANN are unnormalized (predicted) class log-probabilities, that is $f_i = \log \hat{P}(y = i|\mathbf{x})$ [**?**]. By taking exponents and normalizing across possible labels predicted class probabilities are calculated as per below - this is known as the softmax function:

$$\text{softmax}(\mathbf{f})_i = \frac{\exp(f_i)}{\sum_j \exp(f_j)} = q_i \tag{4.4}$$

where $q_i \in [0, 1]$ captures the predicted probability that the input into the ANN belongs to class i. One reason for the softmax layer's popularity is that it is easily compatabile with the cross entropy loss function defined as $L_i = -\sum_i p_i \log q_i$ [?]. In simple image classification tasks where the classes are mutually exclusive we have $p_i = 0 \forall i \neq C$ and $p_i = 1$ for $i = C$ denoting the correct class. Plugging Equation 4.4 into this equation gives:

$$L_i = -\log\left(\frac{\exp(f_C)}{\sum_j \exp(f_j)}\right) = -f_C + \log\left(\sum_j \exp(f_j)\right) \tag{4.5}$$

which is the loss incurred from one observation, and is clearly continuous and differentiable. The sequence of computation leading from model inputs all the way to the output of scalar loss is known as the forward pass. Usually forward pass is computed simultaneously for several inputs, known as a mini-batch, in which case the average loss across the mini-batch is typically used. The aim of training an ANN is based on learnng model weights that minimize some appropriate loss function, such as the cross-entropy loss above. Most supervised deep learning models, including the basic feedforward-network described above, are nowadays trained using the back-propagation algorithm [?] [?]. The main idea of this algorithm is to use the chain rule to decompose the gradient of a loss function so that it can be efficiently passed back through the network. More formally, assume the loss of an ANN is produced by a sequence of m nested operation:

$$L(y_i, x_i) = f^{(m)}(y_i, f^{(m-1)}(\dots f^{(2)}(f^{(1)}(x_i)))) \tag{4.6}$$

where y_i is the real label of the observation, x_i the input data, the different $f^{(i)}$ may for example represent different types of layers. Employing the chain rule recursively, a simple decomposition gives:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial f^{(m)}} \frac{\partial f^{(m)}}{\partial f^{(m-1)}} \cdots \frac{\partial f^{(1)}}{\partial x_i} \tag{4.7}$$

These computations are done in the opposite order of the forward-pass operations; hence back-propagation. Above example is simplistic since usually in addition to inputs from preceding layer, each layer has also its own paramets which can be also be thought as inputs to that layer. The general idea of the chain rule still works in that situation as well, now however the gradient flow bifurcates at each layer; part of gradients flow into tha earlier layer while the others flow back into parameters; this is explained in more depth in Figure 4.2. By representing ANNs as computational graphs, and using the chain rule similar to above, it becomes easy to see how backpropagation can be used to send appropriate gradients to right places even in complex network architectures. Importantly, backpropagation does this efficiently since at each operation all upstream gradients are collated and then passed on to one layer down, which is a lot more efficient than considering every single path through an ANN individually. To see this, consider Figure 4.1: using back-propagation we can start from the output and, for instance, calculate the derivative of the output layer with respect to $H5$ neuron only once, and then pass this derivative to all neurons $H1$ - $H4$ simultaneously, which requires a lot less computation than considering all the paths that involve $H5$ separately.

After gradients are calculated using back-propagation, they are used by an optimization algorithm to change the model's parameters with the aim of minimizing the loss function. Here we consider stochastic gradient descent (SGD) [?] which is likely the most widely used optimization algorithm in deep learning. This algorithm is called stochastic because at each learning iteration only a subset, known as mini-batch, of the data is use to calculate gradients and to perform parameter updates; it has been shown that SGD

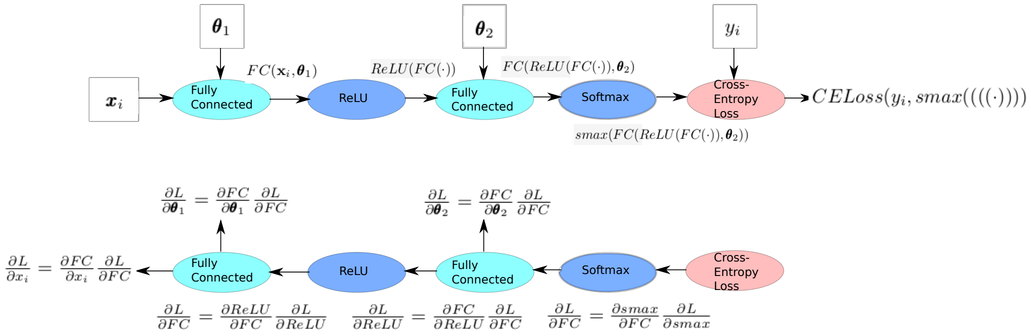
$\boldsymbol{\theta}_1$
$\boldsymbol{\theta}_2$
y_i
$\boldsymbol{x}_i$
Fully Connected
$FC(\mathbf{x}_i, \boldsymbol{\theta}_1)$
ReLU
$ReLU(FC(\cdot))$
Fully Connected
$FC(ReLU(FC(\cdot)), \boldsymbol{\theta}_2)$
Softmax
$smax(FC(ReLU(FC(\cdot)), \boldsymbol{\theta}_2))$
Cross-Entropy Loss
$CELoss(y_i, smax((((\cdot))))$
$\frac{\partial L}{\partial \boldsymbol{\theta}_1} = \frac{\partial FC}{\partial \boldsymbol{\theta}_1} \frac{\partial L}{\partial FC}$
$\frac{\partial L}{\partial \boldsymbol{\theta}_2} = \frac{\partial FC}{\partial \boldsymbol{\theta}_2} \frac{\partial L}{\partial FC}$
$\frac{\partial L}{\partial x_i} = \frac{\partial FC}{\partial x_i} \frac{\partial L}{\partial FC}$
Fully Connected
ReLU
Fully Connected
Softmax
Cross-Entropy Loss
$\frac{\partial L}{\partial FC} = \frac{\partial ReLU}{\partial FC} \frac{\partial L}{\partial ReLU}$
$\frac{\partial L}{\partial ReLU} = \frac{\partial FC}{\partial ReLU} \frac{\partial L}{\partial FC}$
$\frac{\partial L}{\partial FC} = \frac{\partial smax}{\partial FC} \frac{\partial L}{\partial smax}$

Figure 4.2: A simple example of forward (top) and backward passes (bottom) for a simple two-layer ANN, which illustrates how gradients flowing to early layers are calculated by multiplying the local gradient of a given layer with the gradient that flows back from the next layer

converges much faster than calculating gradients always on all available data; the time per update for the algorithm is independet of size of data as long as batch size is held constant [**?**]. The parameter updates are based on moving 'downhill' i.e. in the direction of negative gradients:

$$\boldsymbol{\theta}_{(t+1)} = \boldsymbol{\theta}_{(t)} - \eta \nabla_{\theta_{(t)}} L(\mathbf{X}, \mathbf{y}) \tag{4.8}$$

Most deep learning models are very sensitive to the choice of learning rate η; if it is too high, we are likely to miss minima and conversly there is a risk of local minima and slow training time when the parameter is set too small. Usually learning rate is reduced linearly with training, or in bigger steps at regular intervals, to reduce the impact of noise when we are near a minimum [**?**].

Another common addition to the vanilla SGD is momentum [**?**], which can accelerate learning when there is a lot of noise from SGD or when the Hessian of the loss (matrix of 2nd order derivatives) is ill-conditioned as shown in Figure 4.3. SGD with momentum ammends the original SGD by introducing a velocity term that accumulates gradients from previous iterations with

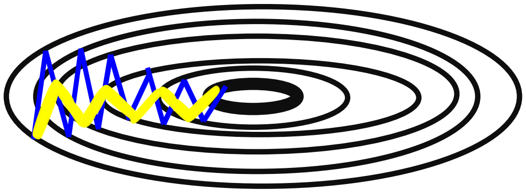

Figure 4.3: Example of ill conditioned Hessian and how momentum speeds up learning by accumulating the gradients of previous iterations such that velocity towards centre (lower loss) is established (shown in yellow). SGD without momentum keeps jumpin across the loss surface as if in a downward sloping canyon, but failing to utilize the slope (blue trace). The contours depict different levels of loss that decreases inwards

an exponential decay. Rumelhart and Hinton [?] described momentum as if dropping a ball-bearing on loss surface and letting momentum drive the ball. Further, the loss landscape can be imagined to be immersed in a liquid with a specified level of viscosity that defines how quickly the momentum fades. Algorithm 1 gives an example of full SGD momentum algorithm for learning model parameters. In practice, back-propagation and optimization can nowadays be done automatically by modern deep learning libraries such as PyTorch [?] and Tensorflow [?].

Algorithm 1 SGD with momentum (following [?])

```
Require: Learning rate $\eta$, Momentum decay parameter $\alpha$
Require: Initial parameters $\boldsymbol{\theta}$, initial velocity $\nu$
  while Convergence not met do
      Sample a minibatch of m observations from training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$
  and their
      Get corresponding targets from training data $\{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$
      Compute gradient for the minibatch: $\mathbf{g} \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(\mathbf{x}_i|\boldsymbol{\theta}), \mathbf{y}_i)$
      Update velocity: $\boldsymbol{\nu} \leftarrow \alpha\boldsymbol{\nu} - \eta\mathbf{g}$
      Updata parameters: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\nu}$
  end while
```

So far we have considered only simple feedforward ANNs with fully connected layers. Most of the ground-breaking accomplishments over the past decade in deep learning have however been achieved with convolutional neural networks (CNNs) [?] of some sort. This is particularly true for computer vision tasks such as image classifiction [?]. In fact, the term deep learning is often used synonymously with CNNs that have a large number of layers.

Unlike fully connected neurons in feedforward networks, convolutional layer has neurons, usually called kernels, which are connected only to some parts of the input it receives from its preceding layer - together many such neurons span the entire input data. Furthermore, there is weight sharing between the kernels to allow for the recognition of a particular feature anywhere in the input space [?]; convolution layers are thus particularly suited for data with locally correlated structures such as images. Consider an input image of 225x225, a typical convolution filter may have size 3x3 and, after having been trained on image data, could have learnt feature mapping that represents particular visual feature such as a vertical edge. This filter is then replicated over the entire 225x225 input range so that the model can detect that particular feature anywhere in the image. Usually each layer has a multitude of such kernels to detect different features in the input data. Mathematically, this feature detection corresponds to the convolution operation between input data X and convolution kernels K, which for discrete 2D data is given as $S(i,j) = (X * K) = (i,j) \sum_m \sum_n X(i-m, j-n)K(m,n)$ [?] where i and j denotes a particular image pixel location and m and n those of the kernel. Figure 4.4 gives a brief toy example to illustrate this process.

The reason why convolutions have proved so effective for image data is that natural visual scenes present strong local correlations - the world we view is not just a collection of randomly ordered pixels. Additionally, these visual features can appear at essentially anywhere in our visual scenes; thus the need for convolution layers to scan across the whole image [?]. Further, CNNs usually have several layers of convolutions on top of each other. When such deep CNNs are trained on image data, the kernels across the different layers usually learn a hierarchy of visual features, such that the early layers

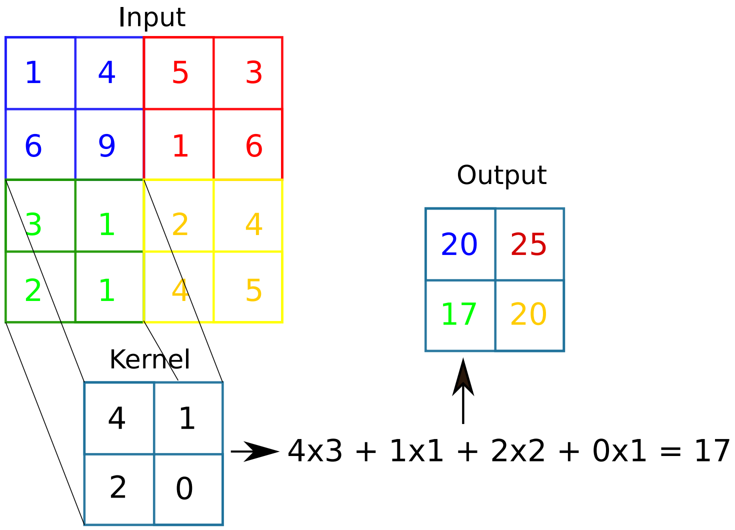

Figure 4.4: A simple example of the computations performed by a single 2x2 convolution kernel on a 4x4 input. Assuming stride length of two squares, the kernel will see each of the four corners and performs a convolution operation on each of them. This is shown explicitly for the bottom left green square. The computation is hence essentially a dot product similarity metric between the kernel and the local image areas.

detect edges and corners, which are then used by middle layers to create contours and parts of objects, and finally later layers build up more complex representations, possibly of complete objects [?]. This shares some similarities with mammals' visual cortex [?] in which earlier (V1) cells respond to edges and bars of different orientations [?] and later ones like V4 and the IT cortext to more complex shapes [?]. This idea of distributed hierarchical representations is crucial to our research. First, the abiliy to learn fundamental visual features, such as edges, should help to generalize models to novel visual scenes. Second, generalization is also improved by hierarchy of features: for example, if a CNN only trained on images of cats and apples, is shown a picture of a dog, it would be more likely to classify it as a cat than apple which is arguably the closer of the two. In the next chapter we show how an extension of this idea can be explotited to perform more accurate predictions for previously unseen classes of objects.

Convolution layers in CNNs are normally followed up by max-pooling functions. Usually this is just the $max()$ function applied to a grid output of its preceding convolution output. For example, in Figure 4.4 the max-pooling would only pass through the value of 25. The benefit of such down-samplign is local invariance: visual objects are not completely rigid and by passing on only the largest value, we are likely to produce the same output even if the values of the square changed around a bit [?] (imagine a letter that's slightly rotated between different views).

One of the most important features of CNNs is that they can be trained end-to-end using back-propagation and SGD. A consequence of paramount importance is that the type of visual features which convolution kernels learn to represent is fully determined by what best fits the data, and they can be learned from raw data. This is in sharp contrast to earlier computer vision feature extraction techniques [?] such as SIFT [?] in which manually designed features are used. Following the breakthrough performance of the seminal AlexNet CNN [?] at the ImageNet 2012 competition, deep CNNs of various designs have become state-of-the-art in virtually all machine vision classification and detection tasks [?].

In addition to convolution and pooling layers, CNNs involve several other important architectural designs. For example, deep networks with a large number of convolution and pooling layers have been found to perform better [?] than shallow ones. This is exploited by several CNN models such as the ResNet architectures which can have more than 100 layers [?]. Theoretically, depth increases the representational capacity of the network [?]. Representational capacity is also increased by the use of ReLUs on top of convolution layers. All in all, modern deep CNNs can easily have hundres of millions of parameters and the ability to learn super-human performance on many image classification tasks [?]. Due to their large representational capactiy, these models can also easily overfit training data and the development of appropriate regularization techniques has played a large role in CNNs recent achievements. Perhaps the most popular regularization technique, and the one applied in this work, is DropOut [?]. During model training, dropout deactivates a each neuron in each iteration with a probability P. The consequence of this is that over the entire training period we essentially end up training an ensemble of different models, and the final model is an average of exponentially many sub-models [?]. This method regularizes the network as individual neurons can not rely too much on any other neurons and the output expected to get out of them. Durng test-time, dropout is turned off.

Many of the theoretical concepts of modern deep CNNs have been around for several decades [?]. The reason for their recent surge in popularity is to a large extent that they have become a lot easier and faster to train as our computers have gotten more powerful and our datasets much larger. In particular, the ability to train deep models on Graphical Processing Units (GPUs) has cut training times by at least an order of magnitude [?]. However, we have only been able reap the benefits of faster compute and better models because of 'big data'. It has been suggested that CNNs with around

5000 labels per training category can on average start to match human performance [**?**] - over the past decade we have seen the advent of datasets that have up to tens of millions of training examples [**?**], [**?**], [**?**].

Despite the many successes of deep learning in various computer vision tasks, most of these rely on well defined training and testing envionments and do not generalize well beyond them; we are thus stil far from general human-level performance. Upon observing almost any visual scene, humans are able to effortlessly cut it up into segments of distinct objects [1], even if we have never seen the scene before. This is a remarkable ability considering that the world we live in can present us with an infinite amount of visual variation to our retinas and yet we are able to easily recognize tens of thousands of distinct object categories [?] with invariance to various factors such as movement, lighting, shade, orientation and partial occlusion [?]. Rosch *et al.* [1] argue that this is because of the non-random, and hierarchical, structure of visual objects, for instance no breed of dogs will have wings but they will often share many features such as four legs with certain type of paws. In general entities that are closer to each other in a hierarchical semantic taxonomy will also typically share more visual features; dogs and birds are still more alike than dogs and vehicles since both belong to a higher level category of animals which all share certain visual attributes.

Standard deep learning models used for image classification do not exploit this semantic taxonomy since they are trained against one-hot encoded target vectors. For example, the ImageNet data set [?] contains a large number of different dog breeds and their training label vectors will have 1 for the dimension a given breed and 0 for all the other possible image classes. It follows that label vectors for different breeds are thus orthogonal and normally a model trained on this data will therefore not accomodate for the fact that the two breeds do infact belong to the same higher level categories of dogs, animals and so forth. The chosen level of hierarchy such one-hot encoded labels represent is also usually arbitrary; should an image be labelled a dog or a labrador retriever? However, optimally both would be taken into account, we argue.

Our research is motivated by above observations. We conjecture that we can help deep learning models to learn much better representations that generalize better by teaching them hierarchical taxonomies. There have recently been a few relevant papers exploring this idea. Wang and Cottrell [?] trained a CNN on two-level hierarchical labels of the ImageNet 2012 data [?] and found that this leads to an improved performance on the standard top-5 classification accuracy. The two levels they use are the 'basic level' and the 'subordinate level'; for example, a 'fox' is the basic level of its subordinate 'red fox'. The 'basic level' concept follows from Rosch *et al.* [1] who identified it as the level at which an object is typically identified by humans at first; afterwards object may also be identified at a lower lever called 'subordinate' level. Another simiar study [?] construct a dataset which mostly has coarse labeled images (similar to the basic level) and only some fine grained examples. The authors show that their custom CNN that uses both hierarchies can predit fine-level labels better than a model that's only trained one fine level. This suggests that the fine-grained model is able to borrow strength from coarse labels and consequently generalize better. Peterson *et al.* extended this line of research by exploring the type of representations learnt by a CNN trained on both 'basic' and 'sub-ordinate' labels. First, the authors found that including the basic-level labels in pre-training or fine-tuning led to a much more clustered representation of the feature spaces extracted from the final layer of the CNN (e.g. the features vectors of different breeds of dogs were now bundeled up together whilst if trained on just subordinate labels, then there was no clustering of similar categories). The features also had learn separate hierarchies for natural and man made objects. Finally, the authors illustrate the generalization power of these representations by running a zero-shot learning experiment in which the model sees only few examples of either sub- or basic-level objects and then tries to find all other images from the data set belonging to the given label. Interestingly, the results show that the supplementary basic-level training has led to a bias to label objects at this level which is congruent with corresponding studies in humans [5].

Despite the important contribution of above works, they are limited in that they only consider two levels of a semantic taxonomy. It has been shown that in reality we posses a much more complex multi-level semantic hierarchy and the level at which we identify visual objects depends various factors such as the time allowed for identification and knowledge of the subject [2]. In this paper we exploit a fully set of hierarchical taxonomy via semantic embeddings built on a large-scale lexical database WordNet [**?**].

Above works illustrate the implicit connction between text and visual data processing in humans. The two were shown to be explicitly linked by Joliceur *et al.*

Mention low accuracy and difficulty of task for zero-shot learnig.

<u>Pictures and Names: Making the Connection [2]</u>

Overview

This extends the work of [1]. The authors find that the basic level is a level at which identification is fastests on average. In fact, it is found that humans will work through taxonomy by first classifying an object at basic level and after that at a subordinate level. The classification we perform is thus time-dependent. Further the authors find that the same process is used to name text and visual data, which implies that there is an important link between our perception of visual scenes and semantic knowledge. More specifically, humans are able to use semantic knowledge to generalize to superordinate categories from seen objects. For example, upon seeing an image of a chair, we can say that this belongs to a broader category of furniture eventhough the image alone doesn't reveal that such higher level of abstraction exists. On the other hand, upon hearing about 'an office chair with wheels' we can easily access our visual memory to imagine what this would look like. Thus there is a strong link between our visual and semantic processes. Another important contribution of this work is that whilst basic level is indeed the level where usually identification is first made, this is not true always, especially when you have atypical examples: penguin for instance would be first identified as penguing rather than a bird which is the basic-level. In general, classification is also constrained by the subject knowledge, for instance a bird-watcher may identify a bird as a Robin but most people will just call it a 'bird'.

Thoughts:

- it is interesting that usually subordinate classification depends on the basic level. DL should thus take account of hierarchical semantic information optimally.
- in general the close relationship between semantic and visual identification corroborates our aim of using semantic embedding.
- one concern is that we are limited by the lowest level of labels in the imagenet labels, however otherwise model training, unlike humans, is not restricted by classification time and ability to recall detailed labels and thus we should be able to incorporate all the possible semantic knowledge

Bibliography

[1] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem, “Basic objects in natural categories,” *Cognitive Psychology*, vol. 8, no. 3, pp. 382–439, 1976.

[2] P. Joliceur, M. A. Gluck, and S. M. Kosslyn, “Pictures and Naming: Making the Connection,” *Cognitive Psychology*, vol. 16, pp. 243–275, 1984.

[3] J. C. Peterson, P. Soulos, A. Nematzadeh, and T. L. Griffiths, “Learning Hierarchical Visual Representations in Deep Neural Networks Using Hierarchical Linguistic Labels,” 2018.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2015.

[5] F. Xu and J. B. Tenenbaum, “Word learning as Bayesian inference.,” *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, pp. 517–522, 2000.