Convolutional Neural Networks for Image Classification and Captioning

Sachin Padmanabhan SACHIN@CS.STANFORD.EDU
Department of Computer Science, Stanford University

Abstract

In this paper, we explore the use of convolutional neural networks (CNNs) for the image classification and image captioning problems. These tasks are extremely important in modern computer vision and have numerous applications. We explore the performance of several deep learning models on the image classification problem. We then use a CNN trained for the image classification problem as a feature extractor to describe a simple CNN–RNN architecture that achieves near state-of-the-art performance on the image captioning task.

1. Introduction

In recent years, convolutional neural networks (CNNs) have revolutionized computer vision. Although they have been around since at least the 1990s, when they were popularized by Yann LeCun's LeNet (LeCun et al., 1989), (LeCun et al., 1998), it has only recently become possible to train these large, complex networks on very large datasets. Since (Steinkrau et al., 2005) showed the value of using GPUs for machine learning, modern GPU computing and parallel computing methods have massively increased the ability to train CNN models, leading to their rise in research and in industry. We are now able to train networks with millions, or even billions, of parameters on extremely large datasets like ImageNet, relatively efficiently.

Image classification is one of the classic problems in computer vision: given an image, identify it as being a member of one of several fixed classes. Image classification is a classic problem partly because of its numerous applications. Autonomous driving requires fast image classification as a particularly critical primitive. In addition, modern social media and photo sharing/storage applications like Facebook and Google Photos use image classification to improve and personalize user experience on their products. The image classification problem is also representative of many common challenges in computer vision such as intra-class variation, occlusion, deformation, scale variation, viewpoint variation, and illumination. Methods that work well for image classification are likely to translate to methods that will improve other key computer vision tasks as well, such as detection, localization, and segmentation.

A prime example of this is image captioning. The image captioning problem is to, given an image, output a sentence description of the image. The image captioning problem is similar to the image classification problem, but more detail is expected and the universe of possibilities is larger. Modern image captioning systems use image classification as a black box system, so better image classification leads to better captioning. The image captioning problem is interesting in its own right because it combines two major fields in artificial intelligence: computer vision and natural language processing. An image captioning system exhibits understanding of both image semantics as well as natural language. In recent years, natural language processing has also been revolutionized by deep learning, in the form of Recurrent Neural Networks (RNNs). Modern solutions to the image captioning problem have the unique distinction of using state-of-the-art models in both fields.

The applications of image captioning are also numerous. Besides similar applications in social media, image captioning has a great number of image-to-speech applications. Such systems could be used to augment user experience in driving, tourism, and many other situations. Image captioning is also very useful as a subcomponent for image search engines. Finally, image captioning systems could become a crucial component of accessibility software in the future, helping humans with vision problems draw meaning from their surroundings.

Project done in Stanford University's CS 231A (Computer Vision) course taught in Spring 2016 by Professor Silvio Savarese.

2. Literature Review

2.1. Previous Work

State-of-the-art CNN models for image classification are judged by their performance on the ImageNet challenge. They have proceeded incrementally since (Krizhevsky et al., 2012) introduced AlexNet in 2012, with modifications to the architecture that have improved performance. In 2014, (Szegedy et al., 2015) introduced GoogLeNet, which was an improvement on AlexNet, mainly through greatly reducing the number of parameters involved. Also, in 2014, (Simonyan & Zisserman, 2014) introduced the VGGNet, which achieved good performance because of the depth of the network. Most recently, in 2015, (He et al., 2015b) introduced the ResNet, which utilizes "skip connections" and batch normalization. The performance of these models on ImageNet is shown in Table 1.

Table 1. Error rates on ImageNet challenge.

Model	Top-5 Error Rate
AlexNet	15.3%
GoogLeNet	6.67%
VGGNet	7.32%
ResNet	5.71%

The image captioning problem has also seen a lot of work in recent years. In the last year, (Karpathy & Fei-Fei, 2015) introduced a method that combines a pre-trained CNN (VGGNet) as a feature extractor, a Markov Random Field (MRF) model for alignment, and an RNN for generating text descriptions. The approach of (Donahue et al., 2015), similarly, uses a pre-trained VGGNet as a feature extractor, and directly inputs these feature vectors and the word embedding vectors for sentences at each time step of a Long Short-Term Memory (LSTM) model, which was introduced by (Hochreiter & Schmidhuber, 1997). In (Vinyals et al., 2015), Vinyals et al. improve on this approach by only inputting the image vector at the first time step of the LSTM, which they found to improve the results.

2.2. Contributions

In this paper, we explore both the image classification problem and the image captioning problem.

For the image classification problem, we explore and motivate methods for improving the classification accuracy of CNN models in a bottom-up manner. Along the way, we show the impact that these modifications have on model performance. We have implemented all the models we tested (plus more popular models), in Python using both the Keras and Lasagne deep learning libraries. This "model zoo" is available for the community to use and improve.

For image captioning, we have implemented a method in the style of Vinyals et al. The method compares to the performance of the original implementation. However, we found that with smaller amounts of training data, using Gated Recurrent Units (GRUs) (Cho et al., 2014) instead of LSTMs improve the performance of the model.

All the source code for this project can be downloaded `here`.

3. Technical Solution

3.1. Overview

In this section we describe the architectures of our model at a high level. We first describe the datasets and technical stack used, as well as the preprocessing we did on the data. We discuss our choice of stochastic gradient descent (SGD) with Nesterov momentum as our optimization method, initial choice of weights, regularization methods, and our choice of rectified linear units (ReLUs) as the nonlinearity applied after layers.

Then, we describe the models that were developed for image classification. We begin with the baseline, which was a simple softmax regression model (or, a single layer neural network). We then move to a multi-layer perceptron network, and then start adding convolutional layers and max-pooling layers in different ways.

For image captioning, we give a description and motivation of the implemented CNN-RNN models and their components.

3.2. Details

3.2.1. Datasets

To train models for image classification, we used the Tiny ImageNet dataset, which was created for CS231N at Stanford University. In the original ImageNet dataset, there are over $1,200,000$ labeled training images, each with an average resolution of 256×256 pixels. Each image is labeled with one of 1000 different classes. The Tiny ImageNet dataset has fewer classes: 200 instead of 1000. The training set has $10,000$ images: 500 from each class, $10,000$ validation images (50 from each class), and $10,000$ test images (50 from each class). Each image has a fixed resolution of 64×64 pixels. The Tiny ImageNet dataset can be downloaded

by following the link `here`.

To compare our results, we also use the MNIST and CIFAR-10 datasets.

The MNIST dataset, introduced by (LeCun et al., 1998) consists of 70,000 labeled examples (60,000 training examples and 10,000 test examples) of handwritten digits (0–9), each of which are 28×28 black-and-white images. The MNIST dataset can be found `here`.

The CIFAR-10 dataset, introduced by (Krizhevsky & Hinton, 2009) consists of 60,000 images (50,000 training examples and 10,000 test examples), each labeled with one of 10 classes. Each image is a 32×32 color image. The CIFAR-10 dataset can be found `here`

We also implicitly use the ImageNet dataset because we also use a pre-trained VGGNet model, which was trained on the ImageNet dataset.

For image captioning, we use the Flickr8k dataset, introduced by (Rashtchian et al., 2010), which consists of 8,000 images (6,000 training images, 1,000 validation images, and 1,000 test images), each of which are paired with a list of 5 different captions, which provide descriptions of the entities and events in the image. The Flickr8k dataset can be downloaded by following the link `here`.

3.2.2. Preprocessing

For image classification, we apply data augmentation by randomly shifting or horizontally flipping images in our dataset. This increases the size of the training set, which will increase accuracy by decreasing overfitting. Although this method was implemented, the training times with our resources were prohibitively long because of the new size of the training set, so we weren't able to get results. Data augmentation should definitely be explored in future work, however.

For image captioning, we preprocessed each image by resizing (and transposing) it to be $3 \times 244 \times 244$, which is the expected input to the VGGNet model. We also preprocessed the given captions by tokenizing, lowercasing, and applying `<BEGIN>` and `<END>` tokens to the beginning and end of the caption, respectively. We padded with `<UNUSED>` tokens until the caption reached the max caption length of 16. To generate the training set, we generated 15 partial captions from each image-caption pair. The training example was then (image, partial caption) $\rightarrow$ (next word). In this way, each image-caption pair generated 15 training examples. Thus, the final preprocessed training dataset had $6000 \cdot 15 = 90000$ training examples.

3.2.3. Technical Stack

Since we required training relatively large models on a relatively large amount of data, we require more numerical computing power than a standard CPU. Thus, we ran our methods on a `g2.2xlarge` GPU instance from Amazon Web Services (AWS).

We also required building and iterating on several models, training them, and analyzing the performance, so we used the Keras deep learning library for Python (although our models are implemented using Lasagne as well). Keras and Lasagne allowed us to build complicated models sequentially with relative ease just by stacking layers, and to also train these models just as easily.

3.2.4. Optimization Algorithms

To find the optimal parameters in our networks, we explored many choices of optimization algorithms, including vanilla stochastic gradient descent (SGD), SGD with Nesterov momentum updates, Adadelta, and RMSprop.

For training CNNs for image classification, we found that with almost all different weight initializations, both Adadelta and RMSprop got stuck in local optima: the training accuracy skyrocketed to almost 100% while the validation accuracy was almost as bad as random guessing (0.05%). We did find that SGD worked, but frequently gave worse results than SGD with Nesterov momentum updates.

For training the final merged RNN network for image captioning, we found identical results: we again found that Adadelta and RMSprop got stuck in local optima – the loss function stopped decreasing after around 5 to 10 epochs of training. Again, vanilla SGD worked but using Nesterov momentum updates gave better convergence.

3.2.5. Weight Initialization

Since neural network objective functions are highly nonconvex, the initial choice of weights for the network is critical, since it is easy to get stuck in local optima.

We experimented with uniform weight initialization, and the recommendations of (Glorot & Bengio, 2010) and (He et al., 2015a) for ReLU neurons.

We found that using Glorot and He initializations led to a lower frequency of getting stuck in local optima than using uniform weight initialization, but the difference between the two was not particularly significant for our experiments. Thus, we used Glorot et al.'s

weight initialization in our final models.

3.2.6. Regularization

To avoid overfitting, especially with complex models like neural networks, it is often critical to incorporate weight regularization into the model. This penalizes large, biased weights in favor of smaller weights that cannot overfit as much to data.

One common method of regularization is to add $\lambda\|W\|^2$ to the network objective function which we minimize, where W is a vector containing all the weights of the network. In practice, this is often implemented by regularizing each layer. We found that this particular method of regularization does not make much of an impact on the validation accuracy, but perhaps it would be worthwhile to include for larger models on larger datasets.

Another common method of regularization, introduced by (Srivastava et al., 2014), is dropout. In dropout, we keep a neuron active during training with probability p, and otherwise set it to 0. We used dropout to control overfitting and reduce training time, particularly on layers that have lots of parameters. The values of p for these layers were tuned by choosing different values on different runs.

3.2.7. Nonlinearities

After each (non-pooling) layer in a CNN, we apply a nonlinearity that maps the score at each neuron to another real number. The traditional nonlinearity is the sigmoid function, $\sigma(x) = 1/(1 + e^{-x})$, which maps a real number x to a value in the interval $[0, 1]$. Another traditional nonlinearity is the tanh function, which has the same shape as the sigmoid function but maps real numbers into the interval $[-1, 1]$.

Recently, the most popular nonlinearity is the rectified linear unit (ReLU). The definition is $f(x) = \max(0, x)$. The popularity of nonlinearity is (somewhat ironically) due to its simple, linear form. Because of this, it greatly improves the speed of SGD compared to sigmoid and tanh functions. Furthermore, the ReLU function is very easy to evaluate, especially because it doesn't involve any exponentials, or even division.

However, ReLUs are fragile and sometimes can permanently die during training. To fix this problem, the leaky ReLU was proposed, which has a small negative slope when for negative inputs.

We found that using ReLU units improves training time as well as accuracy, as expected from results in the literature. However, leaky ReLU units did not give an improvement over ReLU.

3.2.8. Image Classification Models

In this section, we show the architectures for the models we build for image classificaiton.

There are two types of neural network layers that are specific to CNNs. The first is a convolutional layer. This layer runs a sliding window through the image and at each step convolves the subimage with a number of filters, producing a new volume where we have increased the depth. A pooling layer downsamples the volume along the spatial dimensions by a certain factor. This reduces the size of the representation and the number of parameters which makes the model more efficient and prevents overfitting. The most common type of pooling layer is a max-pooling layer, which takes the max from each downsampled block.

In the following descriptions, CONV(n, s) denotes a convolutional layer with n filters and a window size of $s \times s$, POOL(z) denotes a pooling layer with a factor of z, and FC(n) denotes a fully-connected layer with n units.

Softmax (baseline): FC(200), SOFTMAX

MLP: FC(512), RELU, FC(200), SOFTMAX

LeNet: CONV(32, 3), RELU, POOL(2), CONV(32, 3), RELU, POOL(2), FC(512), RELU, FC(200), SOFTMAX

CNN1: CONV(32, 3), RELU, CONV(32, 3), RELU, POOL(2), DROPOUT(0.25), FC(128), RELU, DROPOUT(0.5), FC(200), SOFTMAX

CNN2: CONV(32, 3), RELU, CONV(32, 3), RELU, POOL(2), DROPOUT(0.25), CONV(64, 3), RELU, CONV(64, 3), RELU, POOL(2), DROPOUT(0.25), FC(512), RELU, DROPOUT(0.5), FC(200), SOFTMAX

CNN3: CONV(16, 5), RELU, POOL(2), CONV(16, 3), RELU, POOL(2), CONV(32, 3), RELU, FC(200), SOFTMAX

CNN4: CONV(32, 5), RELU, CONV(32, 3), RELU, POOL(2), CONV(64, 3), RELU, POOL(2), CONV(128, 3), RELU, POOL(2), FC(256), RELU, DROPOUT(0.5), FC(200), SOFTMAX

CNN5: CONV(64, 5), RELU, CONV(64, 5), RELU, POOL(2), CONV(64, 3), RELU, CONV(64, 3), RELU, CONV(64, 3), RELU, POOL(2), CONV(128, 3), RELU, FC(256), RELU, DROPOUT(0.5), FC(256), RELU, DROPOUT(0.5), FC(200), SOFTMAX

In addition, we also implemented larger popular models not shown here, including AlexNet, GoogLeNet, and VGGNet.

3.2.9. Image Captioning Model

In this section, we explain the image captioning method implemented, which closely follows that presented by Vinyals et al.

Given an image I and sentence description S in the training set, we choose our parameters by maximizing the probability of a correct description given the image:

$$\begin{aligned}\theta^* &= \arg\max_\theta \prod_{(I,S)} p(S|I;\theta) \\ &= \arg\max_\theta \sum_{(I,S)} \log p(S|I;\theta)\end{aligned}$$

We can then model the probability of each sentence given an image as

$$p(S|I;\theta) = \prod_{t=0}^{N} p(S_t|I, S_0, \ldots, S_{t-1};\theta)$$

It is natural to model this probability with an RNN, where the words we condition on are represented as memory h_t. After we see a new word, the memory is updated as $h_{t+1} = f(h_t, x_t)$. We need only decide what the x_t's will be, as well as the form of f. For the x_t's, we follow Vinyals et al. We append a fully-connected layer of length L to the top of our pretrained CNN feature extractor and feed this into the RNN at time $t = -1$ as x_{-1}. Then, for each subsequent time step (starting with the `<BEGIN>` token at $t = 0$), we feed in the word embedding of length L of the subsequent word in the caption, and the RNN outputs the probabilities of the next word being any of the words in the vocabulary. That is,

$$\begin{aligned}x_{-1} &= \text{CNN}(I) \\ x_t &= W_e S_t, \quad t \in \{0, \ldots, N-1\} \\ p_{t+1} &= \text{RNN}(x_t), \quad t \in \{0, \ldots, N-1\}\end{aligned}$$

We found $L = 128$ to work well and also train relatively quickly.

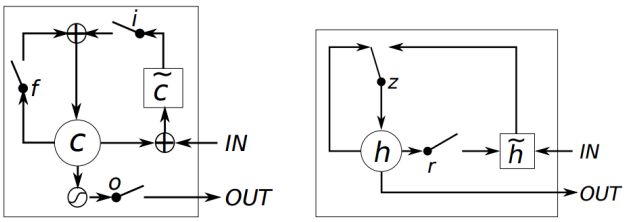
This setup lends itself to the following loss function for each (image, caption) example pair.

$$J(I, S) = -\sum_{t=1}^{N} p_t(S_t)$$

The total loss function is the sum of these over all the training example pairs. The loss function is minimized with respect to all the parameters of the RNN (described below), the fully-connected layer that embeds images, and the word embeddings W_e.

We also have to choose f. In Vinyals et al., a Long Short-Term Memory (LSTM) unit is used. In our implementation, we found this choice to work well when training the model on the entire dataset, but for smaller subsets of the dataset, the Gated Recurrent Unit (GRU) was found to work better. Figure 1 illustrates the difference between these units.

Figure 1. Illustrations of LSTM (left) and GRU (right)



(a) Long Short-Term Memory (b) Gated Recurrent Unit

To predict the caption for an image, sample the model directly; that is, we feed in the image $x_{-1} = \text{CNN}(I)$ at $t = -1$, and the learned word embedding for the `<BEGIN>` token at $t = 0$; that is, $W_e S_0$. Then, we let $S_{t+1} = \arg\max_S \text{RNN}(x_t)$ and stop once we sample the `<END>` token; that is, we choose the highest probability word at each time step. The original model presented in Vinyals et al. uses beam search rather than sampling to construct the caption. Sampling words works reasonably well and is very efficient, but it is definitely worthwhile to implement beam search in future work.

4. Results

4.1. Image Classification

We ran each of the final models described above (after tuning hyperparameters manually) on the Tiny ImageNet dataset for 200 epochs.

For comparison, we also ran each of the models on both the MNIST and CIFAR-10 datasets for 100 epochs, which we found was enough for convergence. The results are shown in Figure 2.

We can see that as we add convolutional layers and increase the depth and complexity of the model, the performance of the model increases as well.

Figure 2. Image classification results.

Model	Tiny ImageNet	CIFAR-10	MNIST
Softmax (baseline)	1.3%	34.5%	92.6%
MLP	3.4%	49.1%	98.2%
LeNet	17.2%	66.4%	99.2%
CNN1	16.5%	66.8%	99.3%
CNN2	20.6%	72.2%	99.6%
CNN3	17.5%	60.7%	99.1%
CNN4	27.2%	64.9%	99.4%
CNN5	32.3%	74.5%	N/A

4.2. Image Captioning

We ran the image captioning model with a pre-trained VGGNet as a feature extractor described above, with both GRUs and LSTMS. For each model, we first trained it on a subset of 600 images from the Flickr8k dataset, and then on the complete training set of 6000 images.

It is clear that RNNs need a lot of data to train, and even this amount of data is not sufficient to achieve completely state-of-the-art results. To match these state-of-the-art models, it is necessary to use the Flickr30k dataset, or the still larger MSCOCO dataset. In future work, this should definitely be done.

However, through our experiments, we found that when using the smaller training set of 600 images, using a GRU gave better captions than using an LSTM, although LSTMs are much more prevalent in the literature. We did validate, however, that given the entire dataset, using the LSTM did result in better captions.

Some example captioning results are shown for the GRU model trained on 600 images in Figure 3 and for the LSTM model trained on 6000 images in Figure 4. We ran a BLEU evaluation for the final LSTM model and found that it achieved a BLEU score of 21.0, which is not quite at the state-of-the-art.

However, more computational resources and power would allow for more extensive hyperparameter tuning, fine-tuning of the VGGNet model, and training for more epochs. This would boost the results to be up to par with current research.

5. Conclusions

The first major conclusion from this project is that deep learning is an extremely powerful tool. The representative power of neural networks is amazing, and they have been specialized for use in computer vision and natural language processing, turning tasks that appear very hard, or even impossible, into reasonable ones.

Another major conclusion is regarding factors that greatly influence the performance of neural networks. We have seen that in CNNs, the depth of the model has a strong correlation with performance. Tuning hyperparameters also proved to boost the performance of CNN models by a good amount.

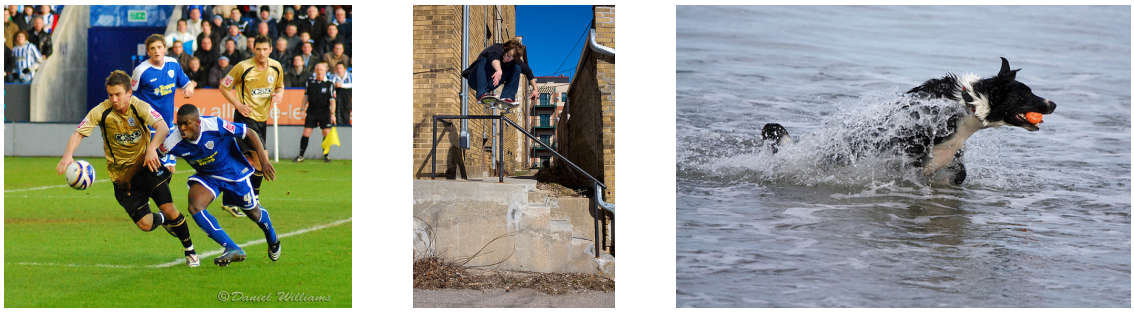
In addition, we have seen many ways that neural networks can overfit, and how to prevent this. Regularization prevents the network from learning large weights. In the form of dropout, this benefits the training time as well. Additionally, neural network models possess highly nonconvex objective functions, so getting stuck in local optima is a real worry. To battle this problem, as we have seen, the choice of optimization method as well as the initial choice of weights is important.

We also noticed that in our particular image captioning application, Gated Recurrent Units (GRUs) perform better than Long Short-Term Memory units (LSTMs) on smaller amounts of data, although LSTMs performed better on larger amounts of data. This is potentially just a coincidence, but it is worthwhile to run more experiments in future work to explore this.

Finally, we have seen that using neural networks with pre-trained weights have a lot of applications. We have seen that using a network trained on ImageNet as a feature extractor is incredibly effective. We have seen that they can be applied to image captioning, but they can also be applied to many more domain-specific tasks like dog breed identification, or retail item classification. With fine-tuning, this approach is even more powerful. However, this was too computationally expensive given our resources, but should be explored in future work.

Figure 3. Sample captions for the GRU model trained on 600 images. From left to right: "the person is going into the water", "two dogs in a field are running to catch a tennis ball", "man playing tennis"

Figure 4. Sample captions for the LSTM model trained on 6000 images. From left to right: "two soccer players are playing soccer", "a skateboarder is doing a trick on a fence", "the dog is running through the water"



6. Acknowledgements

A sincere thanks to Professor Silvio Savarese for the opportunity to do this project. The amount of knowledge I've gained about deep learning through this project makes me consider it a "class-within-a-class". Truly, I learned more doing this project than I have in entire classes.

Also, thanks to Andrej Karpathy and Professor Fei-Fei Li for the Tiny ImageNet dataset, and for the excellent CS231N lecture notes.

References

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Donahue, Jeffrey, Anne Hendricks, Lisa, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2625–2634, 2015.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015b.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Karpathy, Andrej and Fei-Fei, Li. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137, 2015.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images, 2009.

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

LeCun, Yann, Boser, Bernhard, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Rashtchian, Cyrus, Young, Peter, Hodosh, Micah, and Hockenmaier, Julia. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp. 139–147. Association for Computational Linguistics, 2010.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Steinkrau, Dave, Simard, Patrice Y, and Buck, Ian. Using gpus for machine learning algorithms. In *null*, pp. 1115–1119. IEEE, 2005.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, 2015.