Academic Writing from Samples:

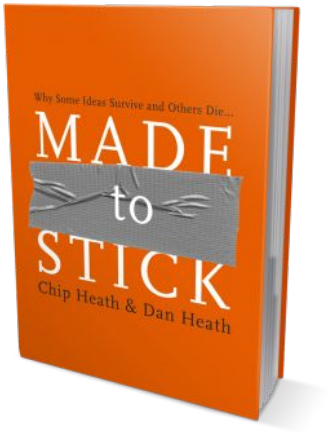
Spike Neural Networks

A guide by Xiaozhe Yao & Yingying Chen

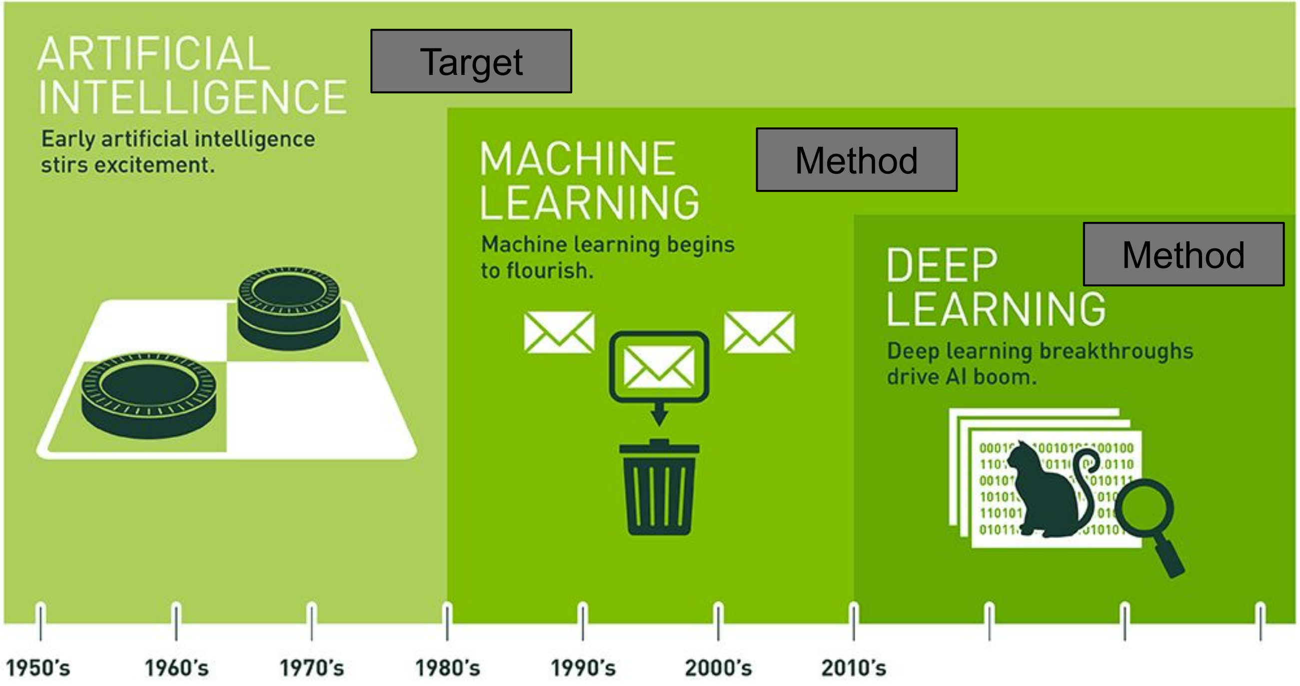
This slides can be downloaded at https://xzyaoi.github.io/teaching/

Contents

- Common Structures for Academic Papers.
- Case Study: Spike Neural Networks.
  - Brief Introduction
  - Analyze 3 papers from INI @ ETH/UZH
  - Writing Skills
- QAs and Sharing



Background of Neural Networks



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

https://www.bilibili.com/video/av359328633

Machine Learning = Find a function from Data

- Speech Recognition

  f() = "How are you?"

- Image Recognition

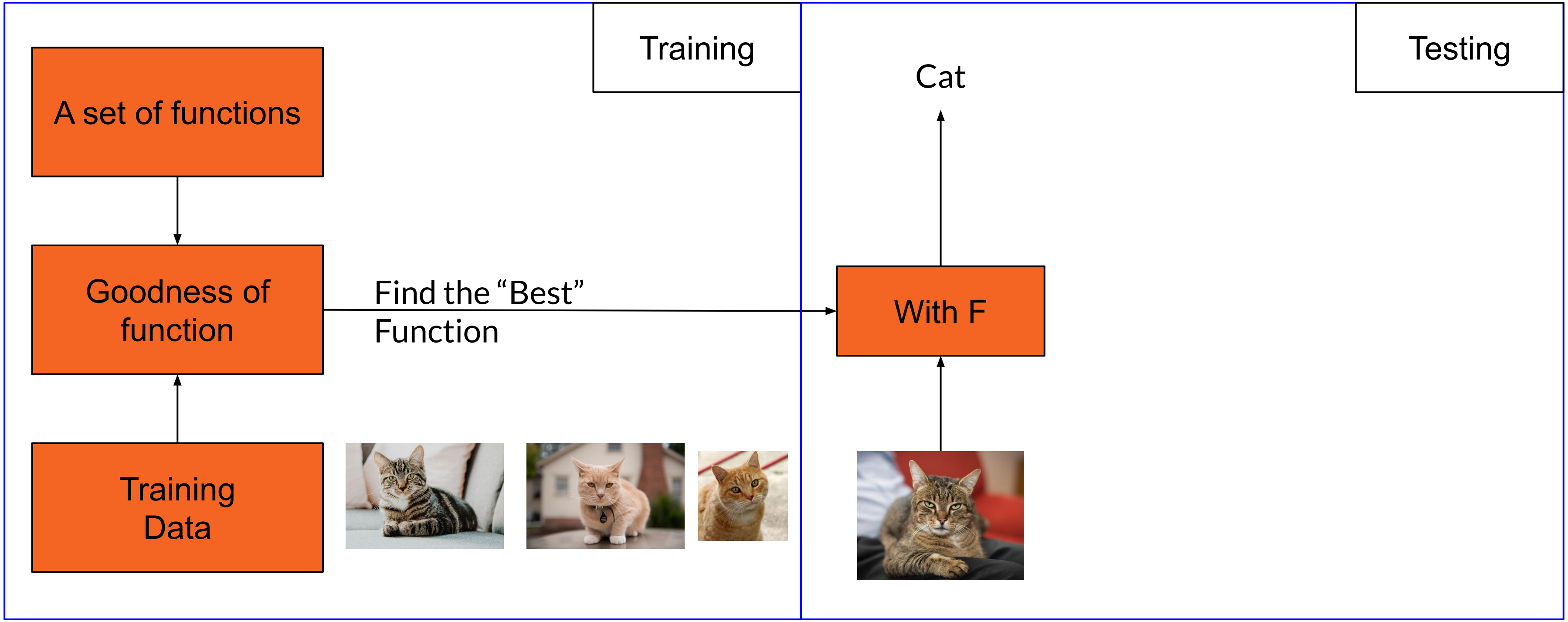
  f() = "cat"

- Play Go

  f() = next move

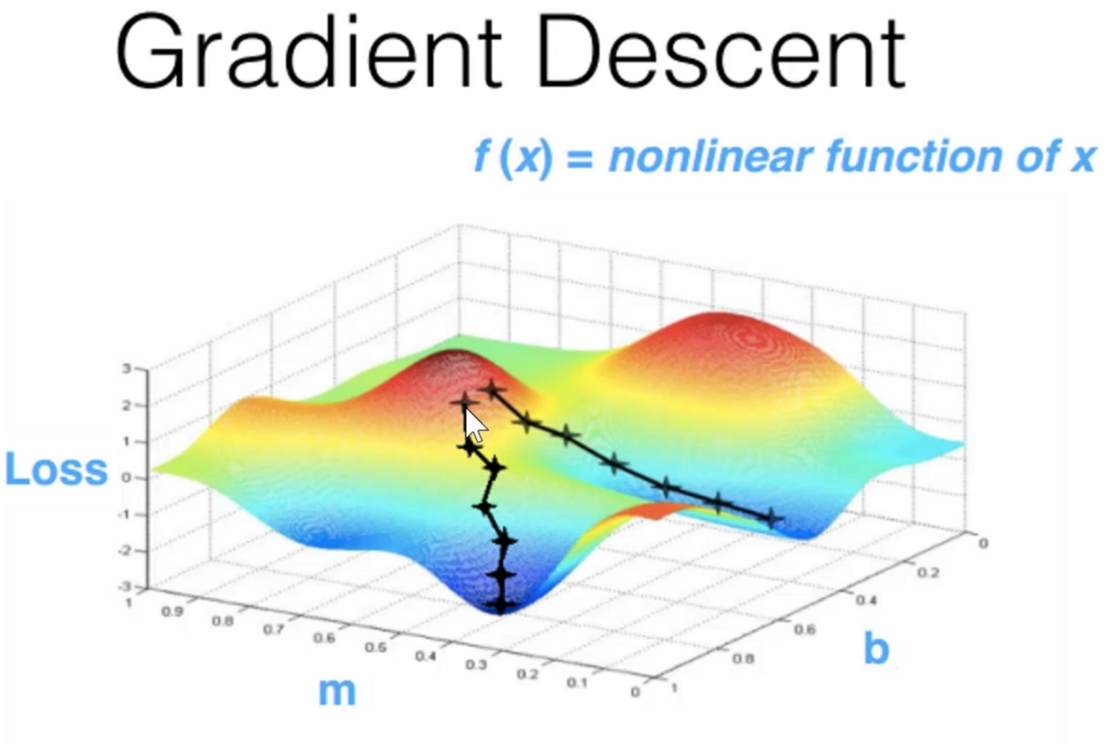
https://www.bilibili.com/video/av359328633

How to find such function?



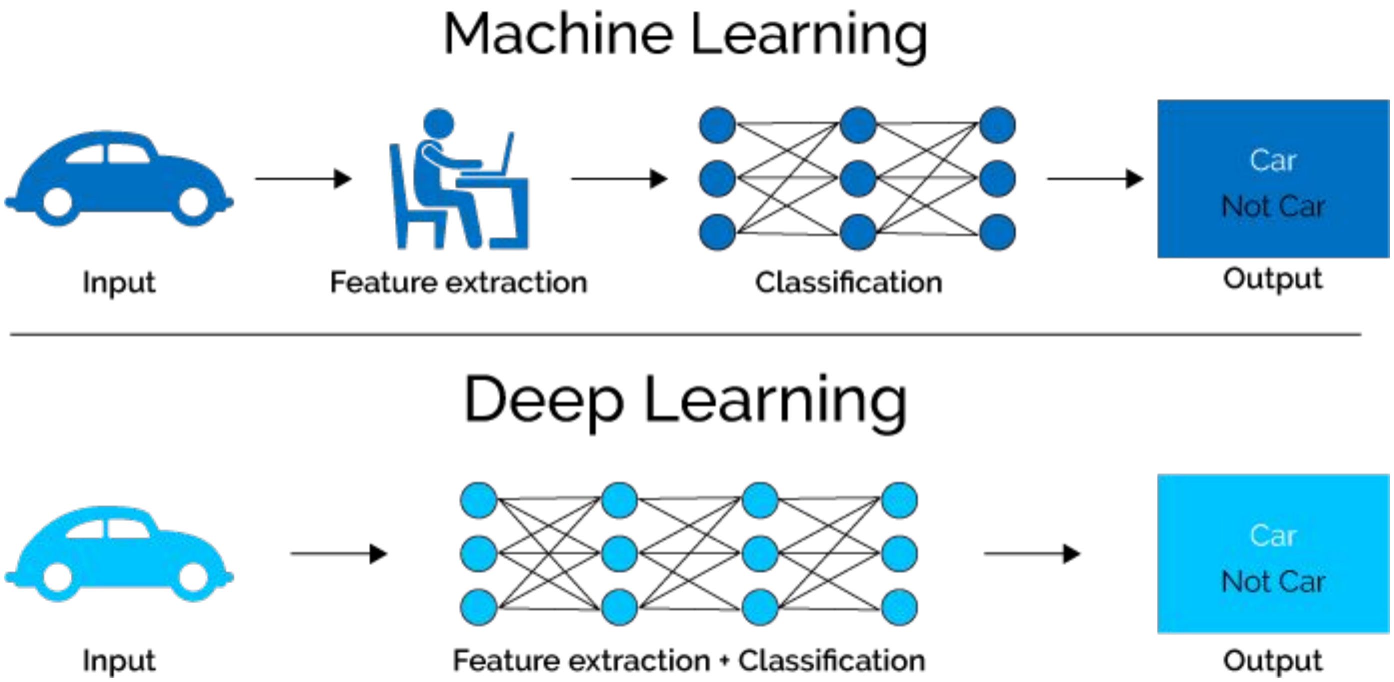
https://www.bilibili.com/video/av359328633

Gradient Descent - Find the best function



Follow the fastest path!
Use ***derivative*** to determine.

What is Neural Networks/Deep Learning?

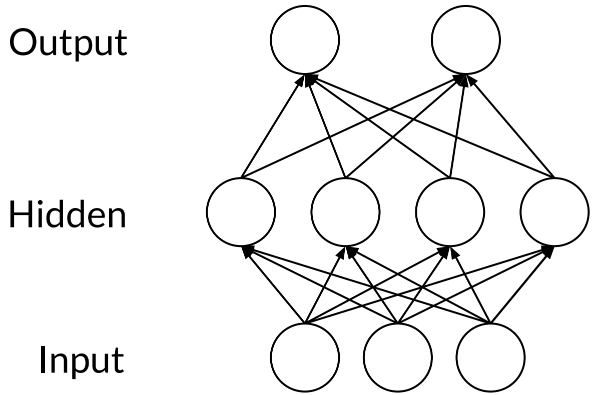


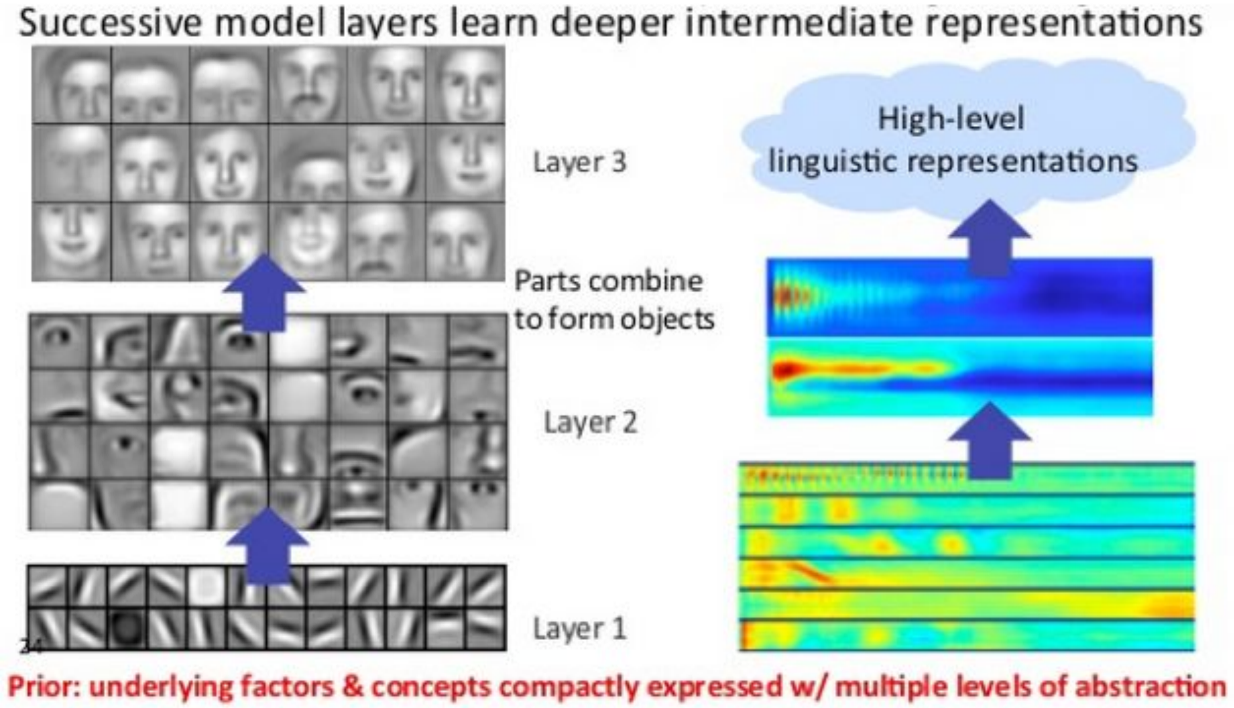
What is Neural Networks/Deep Learning?

Machine Learning:
Size, weight, length, shape…
Other high-level features

Deep Learning:
Just the image!

How deep learning could work that way?





Why Neural Networks?

ImageNet dataset: 14 millions images

Algorithm	Top-1 Error Rate	Top-5 Error Rate
Sparse Coding	47.1%	28.2%
SIFT + FVs	45.7%	25.7%
CNN (Deep Learning)	37.5%	17.0%

ImageNet Classification with Deep Convolutional Neural Networks

By Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Layouts of *ImageNet Classification* Paper

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.

Despite the attractive qualities of CNNs, and despite the relative efficiency of their local architecture, they have still been prohibitively expensive to apply in large scale to high-resolution images. Luckily, current GPUs, paired with a highly-optimized implementation of 2D convolution, are powerful enough to facilitate the training of interestingly-large CNNs, and recent datasets such as ImageNet contain enough labeled examples to train such models without severe overfitting.

The specific contributions of this paper are as follows: we trained one of the largest convolutional neural networks to date on the subsets of ImageNet used in the ILSVRC-2010 and ILSVRC-2012 competitions [2] and achieved by far the best results ever reported on these datasets. We wrote a highly-optimized GPU implementation of 2D convolution and all the other operations inherent in training convolutional neural networks, which we make available publicly[1]. Our network contains a number of new and unusual features which improve its performance and reduce its training time, which are detailed in Section 3. The size of our network made overfitting a significant problem, even with 1.2 million labeled training examples, so we used several effective techniques for preventing overfitting, which are described in Section 4. Our final network contains five convolutional and three fully-connected layers, and this depth seems to be important: we found that removing any convolutional layer (each of which contains no more than 1% of the model's parameters) resulted in inferior performance.

In the end, the network's size is limited mainly by the amount of memory available on current GPUs and by the amount of training time that we are willing to tolerate. Our network takes between five and six days to train on two GTX 580 3GB GPUs. All of our experiments suggest that our results can be improved simply by waiting for faster GPUs and bigger datasets to become available.

2 The Dataset

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

ILSVRC-2010 is the only version of ILSVRC for which the test set labels are available, so this is the version on which we performed most of our experiments. Since we also entered our model in the ILSVRC-2012 competition, in Section 6 we report our results on this version of the dataset as well, for which test set labels are unavailable. On ImageNet, it is customary to report two error rates: top-1 and top-5, where the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model.

ImageNet consists of variable-resolution images, while our system requires a constant input dimensionality. Therefore, we down-sampled the images to a fixed resolution of 256×256. Given a rectangular image, we first rescaled the image such that the shorter side was of length 256, and then cropped out the central 256×256 patch from the resulting image. We did not pre-process the images in any other way, except for subtracting the mean activity over the training set from each pixel. So we trained our network on the (centered) raw RGB values of the pixels.

3 The Architecture

The architecture of our network is summarized in Figure 2. It contains eight learned layers — five convolutional and three fully-connected. Below, we describe some of the novel or unusual features of our network's architecture. Sections 3.1-3.4 are sorted according to our estimation of their importance, with the most important first.

[1] http://code.google.com/p/cuda-convnet/

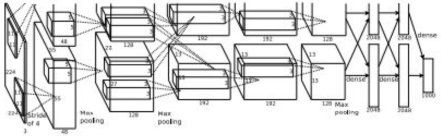


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

4 Reducing Overfitting

Our neural network architecture has 60 million parameters. Although the 1000 classes of ILSVRC make each training example impose 10 bits of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.

4.1 Data Augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations (e.g., [25, 4, 5]). We employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. In our implementation, the transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images. So these data augmentation schemes are, in effect, computationally free.

The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random 224×224 patches (and their horizontal reflections) from the 256×256 images and training our network on these extracted patches[4]. This increases the size of our training set by a factor of 2048, though the resulting training examples are, of course, highly interdependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks. At test time, the network makes a prediction by extracting five 224×224 patches (the four corner patches and the center patch) as well as their horizontal reflections (hence ten patches in all), and averaging the predictions made by the network's softmax layer on the ten patches.

The second form of data augmentation consists of altering the intensities of the RGB channels in training images. Specifically, we perform PCA on the set of RGB pixel values throughout the ImageNet training set. To each training image, we add multiples of the found principal components,

[4] This is the reason why the input images in Figure 2 are $224 \times 224 \times 3$-dimensional.

Title
Abstract

Introduction

The Dataset
The Architecture
Reducing Overfitting

Layouts of *ImageNet Classification* Paper

with magnitudes proportional to the corresponding eigenvalues times a random variable drawn from a Gaussian with mean zero and standard deviation 0.1. Therefore to each RGB image pixel $I_{xy} = [I^R_{xy}, I^G_{xy}, I^B_{xy}]^T$ we add the following quantity:

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

where $\mathbf{p}_i$ and λ_i are ith eigenvector and eigenvalue of the 3×3 covariance matrix of RGB pixel values, respectively, and α_i is the aforementioned random variable. Each α_i is drawn only once for all the pixels of a particular training image until that image is used for training again, at which point it is re-drawn. This scheme approximately captures an important property of natural images, namely, that object identity is invariant to changes in the intensity and color of the illumination. This scheme reduces the top-1 error rate by over 1%.

4.2 Dropout

Combining the predictions of many different models is a very successful way to reduce test errors [1, 3], but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently-introduced technique, called "dropout" [10], consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are "dropped out" in this way do not contribute to the forward pass and do not participate in back-propagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. At test time, we use all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks.

We use dropout in the first two fully-connected layers of Figure 2. Without dropout, our network exhibits substantial overfitting. Dropout roughly doubles the number of iterations required to converge.

5 Details of learning

Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

We trained our models using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005. We found that this small amount of weight decay was important for the model to learn. In other words, weight decay here is not merely a regularizer: it reduces the model's training error. The update rule for weight w was

$$\begin{aligned} v_{i+1} &:= 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i} \\ w_{i+1} &:= w_i + v_{i+1} \end{aligned}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$ is the average over the ith batch D_i of the derivative of the objective with respect to w, evaluated at w_i.

We initialized the weights in each layer from a zero-mean Gaussian distribution with standard deviation 0.01. We initialized the neuron biases in the second, fourth, and fifth convolutional layers, as well as in the fully-connected hidden layers, with the constant 1. This initialization accelerates the early stages of learning by providing the ReLUs with positive inputs. We initialized the neuron biases in the remaining layers with the constant 0.

We used an equal learning rate for all layers, which we adjusted manually throughout training. The heuristic which we followed was to divide the learning rate by 10 when the validation error rate stopped improving with the current learning rate. The learning rate was initialized at 0.01 and

reduced three times prior to termination. We trained the network for roughly 90 cycles through the training set of 1.2 million images, which took five to six days on two NVIDIA GTX 580 3GB GPUs.

6 Results

Our results on ILSVRC-2010 are summarized in Table 1. Our network achieves top-1 and top-5 test set error rates of **37.5%** and **17.0%**[5]. The best performance achieved during the ILSVRC-2010 competition was 47.1% and 28.2% with an approach that averages the predictions produced from six sparse-coding models trained on different features [2], and since then the best published results are 45.7% and 25.7% with an approach that averages the predictions of two classifiers trained on Fisher Vectors (FVs) computed from two types of densely-sampled features [24].

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

We also entered our model in the ILSVRC-2012 competition and report our results in Table 2. Since the ILSVRC-2012 test set labels are not publicly available, we cannot report test error rates for all the models that we tried. In the remainder of this paragraph, we use validation and test error rates interchangeably because in our experience they do not differ by more than 0.1% (see Table 2). The CNN described in this paper achieves a top-5 error rate of 18.2%. Averaging the predictions of five similar CNNs gives an error rate of 16.4%. Training one CNN, with an extra sixth convolutional layer over the last pooling layer, to classify the entire ImageNet Fall 2011 release (15M images, 22K categories), and then "fine-tuning" it on ILSVRC-2012 gives an error rate of 16.6%. Averaging the predictions of two CNNs that were pre-trained on the entire Fall 2011 release with the aforementioned five CNNs gives an error rate of **15.3%**. The second-best contest entry achieved an error rate of 26.2% with an approach that averages the predictions of several classifiers trained on FVs computed from different types of densely-sampled features [7].

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Finally, we also report our error rates on the Fall 2009 version of ImageNet with 10,184 categories and 8.9 million images. On this dataset we follow the convention in the literature of using half of the images for training and half for testing. Since there is no established test set, our split necessarily differs from the splits used by previous authors, but this does not affect the results appreciably. Our top-1 and top-5 error rates on this dataset are **67.4%** and **40.9%**, attained by the net described above but with an additional, sixth convolutional layer over the last pooling layer. The best published results on this dataset are 78.1% and 60.9% [19].

6.1 Qualitative Evaluations

Figure 3 shows the convolutional kernels learned by the network's two data-connected layers. The network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs. Notice the specialization exhibited by the two GPUs, a result of the restricted connectivity described in Section 3.5. The kernels on GPU 1 are largely color-agnostic, while the kernels on on GPU 2 are largely color-specific. This kind of specialization occurs during every run and is independent of any particular random weight initialization (modulo a renumbering of the GPUs).

[5] The error rates without averaging predictions over ten patches as described in Section 4.1 are 39.0% and 18.3%.

7 Discussion

Our results show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results.

To simplify our experiments, we did not use any unsupervised pre-training even though we expect that it will help, especially if we obtain enough computational power to significantly increase the size of the network without obtaining a corresponding increase in the amount of labeled data. Thus far, our results have improved as we have made our network larger and trained it longer but we still have many orders of magnitude to go in order to match the infero-temporal pathway of the human visual system. Ultimately we would like to use very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images.

References

[1] R.M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[2] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. www.image-net.org/challenges. 2010.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *Arxiv preprint arXiv:1202.2745*, 2012.

[5] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification. *Arxiv preprint arXiv:1102.0183*, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. *ILSVRC-2012*, 2012. URL http://www.image-net.org/challenges/LSVRC/2012/.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

Title
Abstract

Introduction

The Dataset
The Architecture
Reducing Overfitting
Details of Learning

Results
Discussion

References

Analysis of the layout - Meta Information

Title
Abstract → Meta Information

Introduction

The Dataset
The Architecture
Reducing Overfitting
Details of Learning
Results

Discussion

References → Meta Information

Meta Information helps reader to identify if the paper is worth reading. It basically includes a very brief introduction (which is about 1~2 min reading time).

As an author, you need to grab the reader's eyes in this part.

Use short sentences, clear words to introduce:

1. What's the problem
2. The most important contribution of your paper.
3. Why it is important.

Spike Neural Networks

Mimic from Human Brain.
Operate using spikes - discrete events.
Discrete events/values is not differentiable.

Examples of Abstract

Deep spiking neural networks (SNNs) hold great potential for improving the latency and energy efficiency of deep neural networks through event-based computation. *However, training such networks is difficult due to the non-differentiable nature of asynchronous spike events*. In this paper, we introduce a novel technique, which treats the membrane potentials of spiking neurons as differentiable signals, where discontinuities at spike times are only considered as noise. This enables an error backpropagation mechanism for deep SNNs, which works directly on spike signals and membrane potentials. Thus, compared with previous methods relying on indirect training and conversion, our technique has the potential to capture the statics of spikes more precisely. Our novel framework outperforms all previously reported results for SNNs on the permutation invariant MNIST benchmark, as well as the N-MNIST benchmark recorded with event-based vision sensors.

Lead In

The problem

What's the paper about

Why it is important

[Lee, Jun Haeng, Tobi Delbruck, and Michael Pfeiffer. "Training deep spiking neural networks using backpropagation." *Frontiers in neuroscience* 10 (2016): 508.]

Examples of Abstract

In order to understand how the mammalian neocortex is performing computations, two things are necessary; we need to have a good understanding of the available neuronal processing units and mechanisms, and we need to gain a better understanding of how those mechanisms are combined to build functioning systems. Therefore, in recent years there is an increasing interest in how spiking neural networks (SNN) can be used to perform complex computations or solve pattern recognition tasks. However, it remains a challenging task to design SNNs which use biologically plausible mechanisms (especially for learning new patterns), since most such SNN architectures rely on training in a rate-based network and subsequent conversion to a SNN. We present a SNN for digit recognition which is based on mechanisms with increased biological plausibility, i.e., conductance-based instead of current-based synapses, spike-timing-dependent plasticity with time-dependent weight change, lateral inhibition, and an adaptive spiking threshold. Unlike most other systems, we do not use a teaching signal and do not present any class labels to the network. Using this unsupervised learning scheme, our architecture achieves 95% accuracy on the MNIST benchmark, which is better than previous SNN implementations without supervision. The fact that we used no domain-specific knowledge points toward the general applicability of our network design. Also, the performance of our network scales well with the number of neurons used and shows similar performance for four different learning rules, indicating robustness of the full combination of mechanisms, which suggests applicability in heterogeneous biological neural networks.

Diehl, Peter U., and Matthew Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity." *Frontiers in computational neuroscience* 9 (2015): 99.

Examples of Abstract

...we need to have a good understanding of the available neuronal processing units and mechanisms, and we need to gain a better understanding of how those mechanisms are combined to build functioning systems...

...it remains a challenging task to design SNNs which use biologically plausible mechanisms...

What's the problem

...We present an SNN for digit recognition which is based on mechanisms with increased biological plausibility...

What's the paper about

...our architecture achieves 95% accuracy on the MNIST benchmark, which is better than previous SNN implementations without supervision...

...the performance of our network scales well with the number of neurons used and shows similar performance for four different learning rules, indicating robustness of the full combination of mechanisms...

Why it is important

Examples of Abstract

Spiking neural networks (SNNs) can potentially offer an efficient way of doing inference because the neurons in the networks are sparsely activated and computations are event-driven. Previous work showed that simple continuous-valued deep Convolutional Neural Networks (CNNs) can be converted into accurate spiking equivalents. These networks did not include certain common operations such as max-pooling, softmax, batch-normalization and Inception-modules. This paper presents spiking equivalents of these operations therefore allowing conversion of nearly arbitrary CNN architectures. We show conversion of popular CNN architectures, including VGG-16 and Inception-v3, into SNNs that produce the best results reported to date on MNIST, CIFAR-10 and the challenging ImageNet dataset. SNNs can trade off classification error rate against the number of available operations whereas deep continuous-valued neural networks require a fixed number of operations to achieve their classification error rate. From the examples of LeNet for MNIST and BinaryNet for CIFAR-10, we show that with an increase in error rate of a few percentage points, the SNNs can achieve more than 2x reductions in operations compared to the original CNNs. This highlights the potential of SNNs in particular when deployed on power-efficient neuromorphic spiking neuron chips, for use in embedded applications.

Analysis of the layout - Lead In and Background

Title
Abstract

Introduction
Related Work* → Lead In & Background

The Dataset
The Architecture
Reducing Overfitting
Details of Learning
Results

Discussion

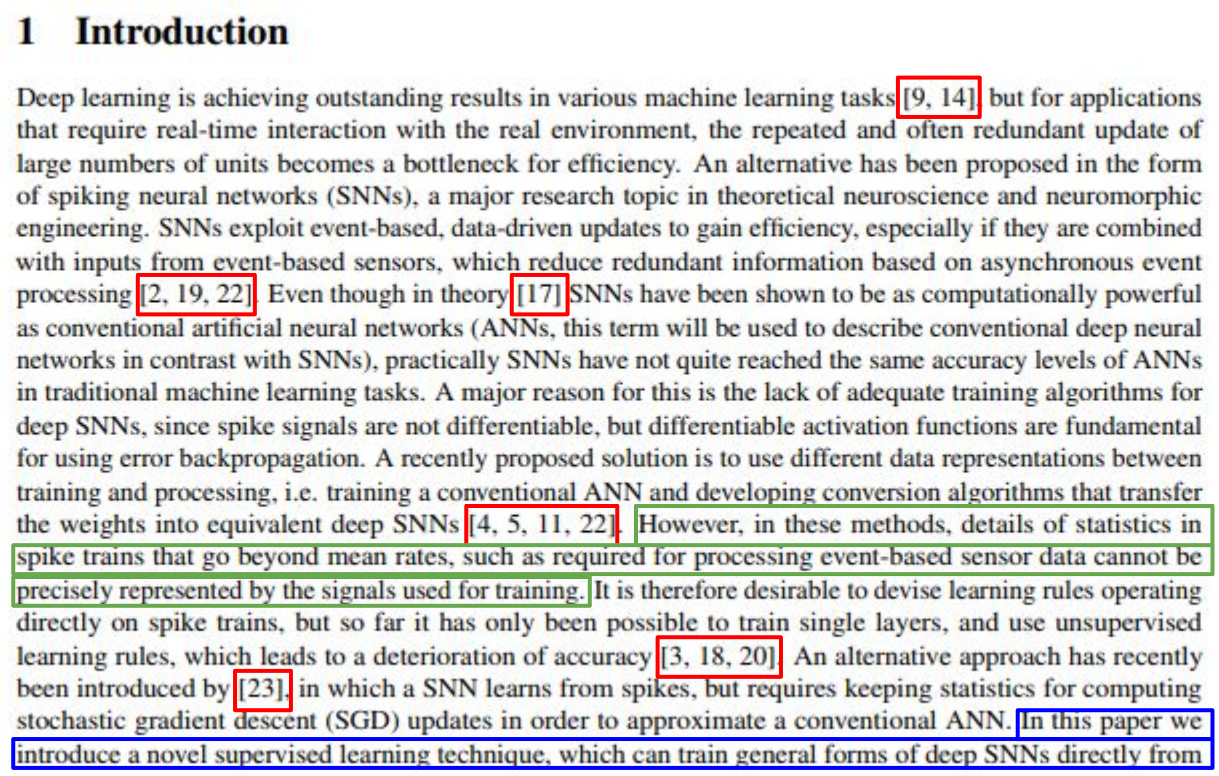
References

Your title and abstract seems impressive, and has attracted readers into your paper (hopefully other than your reviewer :D).

You now have time to:

1. Introduce the whole background, what others have done in the past especially their drawbacks.
2. Demonstrate your advantages over others work.
3. *If possible, introduce how your paper is organized.*

Analysis of the layout - Lead In and Background



1 Introduction

Deep learning is achieving outstanding results in various machine learning tasks [9, 14], but for applications that require real-time interaction with the real environment, the repeated and often redundant update of large numbers of units becomes a bottleneck for efficiency. An alternative has been proposed in the form of spiking neural networks (SNNs), a major research topic in theoretical neuroscience and neuromorphic engineering. SNNs exploit event-based, data-driven updates to gain efficiency, especially if they are combined with inputs from event-based sensors, which reduce redundant information based on asynchronous event processing [2, 19, 22]. Even though in theory [17] SNNs have been shown to be as computationally powerful as conventional artificial neural networks (ANNs, this term will be used to describe conventional deep neural networks in contrast with SNNs), practically SNNs have not quite reached the same accuracy levels of ANNs in traditional machine learning tasks. A major reason for this is the lack of adequate training algorithms for deep SNNs, since spike signals are not differentiable, but differentiable activation functions are fundamental for using error backpropagation. A recently proposed solution is to use different data representations between training and processing, i.e. training a conventional ANN and developing conversion algorithms that transfer the weights into equivalent deep SNNs [4, 5, 11, 22]. However, in these methods, details of statistics in spike trains that go beyond mean rates, such as required for processing event-based sensor data cannot be precisely represented by the signals used for training. It is therefore desirable to devise learning rules operating directly on spike trains, but so far it has only been possible to train single layers, and use unsupervised learning rules, which leads to a deterioration of accuracy [3, 18, 20]. An alternative approach has recently been introduced by [23], in which a SNN learns from spikes, but requires keeping statistics for computing stochastic gradient descent (SGD) updates in order to approximate a conventional ANN. In this paper we introduce a novel supervised learning technique, which can train general forms of deep SNNs directly from

Analysis of the layout - Lead In and Background

spike signals. This includes SNNs with leaky membrane potential and spiking winner-takes-all (WTA) circuits. The key idea of our approach is to generate a continuous and differentiable signal on which SGD can work, using low-pass filtered spiking signals added onto the membrane potential and treating abrupt changes of the membrane potential as noise during error backpropagation. Additional techniques are presented that address particular challenges of SNN training: spiking neurons typically require large thresholds to achieve stability and reasonable firing rates, but this may result in many "dead" neurons, which do not participate in the optimization during training. Novel regularization and normalization techniques are presented, which contribute to stable and balanced learning. Our techniques lay the foundations for closing the performance gap between SNNs and ANNs, and promote their use for practical applications.

Analysis of the layout - Lead In and Background

2 Related Work

Gradient descent methods for SNNs have not been deeply investigated because of the non-differentiable nature of spikes. The most successful approaches to date have used indirect methods, such as training a network in the continuous rate domain and converting it into a spiking version. O'Connor et al. pioneered this area by training a spiking deep belief network (DBN) based on the Siegert event-rate approximation model [22], but only reached accuracies around 94.09% for the MNIST hand written digit classification task. Hunsberger and Eliasmith used the softened rate model for leaky integrate and fire (LIF) neurons [11], training an ANN with the rate model and converting it into a SNN consisting of LIF neurons. With the help of pre-training based on denoising autoencoders they achieved 98.6% in the permutation-invariant (PI) MNIST task. Diehl et al. [4] trained deep neural networks with conventional deep learning techniques and additional constraints necessary for conversion to SNNs. After the training units were converted into spiking neurons and the performance was optimized by normalization of weight parameters, yielding 98.64% accuracy in the PI MNIST task. Esser et al. [5] used a differentiable probabilistic spiking neuron model for training and statistically sampled the trained network for deployment. In all of these methods, training was performed indirectly using continuous signals, which may not capture important statistics of spikes generated by sensors used during processing time. Even though SNNs are optimally suited for processing signals from event-based sensors such as the Dynamic Vision Sensor (DVS) [16], the previous SNN training models require to get rid of time information and generate image frames from the event streams. Instead, we use the same signal format for training and processing deep SNNs, and can thus train SNNs directly on spatio-temporal event streams. This is demonstrated on the neuromorphic N-MNIST benchmark dataset [24], outperforming all previous attempts that ignored spike

Why Citing is important in academic Writing

The importance, or otherwise, of lyrics in popular music, and academic approaches to song lyrics, is subject to much debate. The supposed ‘poor’ standard or presumed meaninglessness of popular music lyrics, become a means to critique popular music. Conversely, it could be argued that too much attention is given to a song’s lyrics, to the point where the music itself is overlooked; it is also possible to overestimate the degree to which the music listener actually listens to the words, or perceives them to be the site of meaning in a song. Nonetheless, Simon Frith suggests that lyrics do allow songs to be ‘used in particular ways’: lyrics facilitate certain ‘creative articulations’. In the case of protest music, the lyrics allow a song to be made to speak to political issues.

The importance, or otherwise, of lyrics in popular music, and academic approaches to song lyrics, is subject to much debate (Frith, 1998; Shepherd, 1999; Fornas, 2003). The supposed ‘poor’ standard or presumed meaninglessness of popular music lyrics, become a means to critique popular music. Conversely, it could be argued that too much attention is given to a song’s lyrics, to the point where the music itself is overlooked; it is also possible to overestimate the degree to which the music listener actually listens to the words, or perceives them to be the site of meaning in a song (Shepherd, 1999:172). Nonetheless, Simon Frith suggests that lyrics do allow songs to be ‘used in particular ways’ (cited in Martin, 1995:273): lyrics facilitate certain ‘creative articulations’ (Johnson, 2000). In the case of protest music, the lyrics allow a song to be made to speak to political issues.

https://student.unsw.edu.au/why-referencing-important

Why Citing is important in academic Writing

1. To show your reader you've done proper research by listing sources you used to get your information.
2. To be a responsible scholar by giving credit to other researchers and acknowledging their ideas.
3. To avoid plagiarism by quoting words and ideas used by other authors.
4. To allow your reader to track down the sources you used by citing them accurately in your paper by way of footnotes, a bibliography or reference list.

You **must** cite:

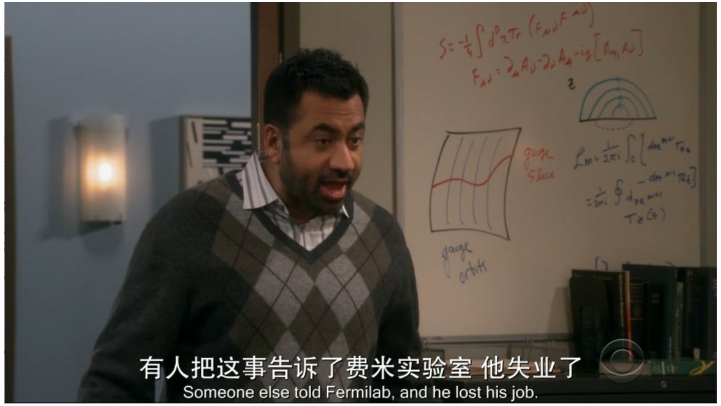
1. Facts, figures, ideas, or other information that is not common knowledge
2. Ideas, words, theories, or exact language that another person used in other publications
3. Publications that must be cited include: books, book chapters, articles, web pages, theses, etc.
4. Another person's exact words should be quoted and cited to show proper credit

When in doubt, be safe and cite your source!

https://libguides.mit.edu/citing The Big Bang Theory S12E21

About Plagiarism

Plagiarism is a very serious offense. If it is found that you have plagiarized -- deliberately or inadvertently -- you may face serious consequences. **In some instances, plagiarism has meant that students have had to leave the institutions where they were studying.**



Analysis of the layout - Lead In and Background

The mammalian neocortex offers an unmatched pattern recognition performance given a power consumption of only 10–20 watts (Javed et al., 2010). Therefore, it is not surprising that the currently most popular models in machine learning, artificial neural networks (ANN) or deep neural networks (Hinton and Salakhutdinov, 2006), are inspired by features found in biology. However, this comparison should be taken with a grain of salt since, despite the biological inspiration, those models use mechanisms for learning and inference which are fundamentally different from what is actually observed in biology. While ANNs rely on 32 bit or even 64 bit messages being sent between units, the neocortex uses spikes, akin to 1 bit precision (if the possible influence of spike-timing on the transmitted message is omitted). Additionally, ANN units are usually perfect integrators with a non-linearity applied after integration, which is not true for real neurons. Instead neocortical neurons are rather leaky integrators, and they use conductance-based synapses which means the change of the membrane voltage due to a spike depends on the current membrane voltage. Another non-biological aspect of ANNs is the type of learning. In ANNs the standard training method is backpropagation (Rumelhart et al., 1985), where after presenting an input example, each neuron receives its specific error signal which is used to update the weight matrix. It seems unlikely that such a neuron-specific error signal would be implemented in the brain (O'Reilly and Munakata, 2000), instead evidence is more pointing toward unsupervised learning methods like spike-timing-dependent plasticity (STDP) (Bi and Poo, 1998), which could be modulated by a global reward signal and therefore could be also used for reinforcement learning.

On the other end of the spectrum, many models in computational neuroscience are modeling biological properties very well but often they are not large scale *functional* systems. However, understanding the computational principles of the neocortex needs both aspects, the biological plausibility and good performance on pattern recognition tasks. If we only focus on biological plausibility, even if we are able to develop functional systems, it is difficult to know which mechanisms are necessary for the computation, i.e., being able to copy the system does not necessarily lead to understanding. Similarly, if we focus only on good performance we will create systems that are working well but which also do not lead to a better understanding since they are too abstract to compare them to the computational primitives of real brains. In recent years many models were developed for pattern recognition tasks that use more biologically plausible mechanisms, marrying both approaches of understanding. One popular approach is to still rely on backpropagation training but afterwards converting the ANN into a spiking neural network (SNN), which we will call "rate-based learning" (Merolla et al., 2011; O'Connor et al., 2013; Hussain et al., 2014; Neil and Liu, 2014; Diehl et al., 2015). While they show very good performance on tasks like the classical machine learning benchmark MNIST (LeCun et al., 1998), this rate-based learning is not very biologically plausible or is at least very much abstracted from the biological mechanism. Other spike-based learning methods often rely on different variants of models of STDP (Brader et al., 2007; Habenschuss et al., 2012; Beyeler et al., 2013; Querlioz et al., 2013; Zhao et al., 2014), providing a closer match to biology for the learning procedure. However, most of those models rely on a teaching signal which provides every single neuron that is used for classification with feedback indicating the correct response, which shifts the problem to the "supervisor neurons" that already need to know the solution. Also, commonly they use features in the neuron/synapse models which make learning easier but are not necessarily biologically plausible; examples include STDP models with highly application specific parameter tuning or current-based synapses, both of which often do not use graded weight changes or graded currents that are observed experimentally.

Here we present a s[illegible]k which relies on a combination of biologically plausible mechanisms and which uses unsupervised learning, i.e., the weights of the network learn the structure of the input examples without using labels. It uses an architecture similar to the one presented in Querlioz et al. (2013), i.e., it uses leaky-integrate-and-fire (LIF) neurons, STDP, lateral inhibition and intrinsic plasticity. However, here we use more biologically plausible components like conductance-based synapses and different STDP rules, all with an exponential time dependence of the weight change. The possibility to vary the design of the learning rule shows the robustness of the used combination of mechanisms. We are training the network on the MNIST dataset without any preprocessing of the data (besides the necessary conversion of the intensity images to spike-trains). The performance of this approach scales well with the number of neurons in the network, and achieves an accuracy of 95% using 6400 learning neurons. Varying the learning rules but keeping the other mechanisms fixed not only shows the robustness of the framework but it also helps to better understand the relationship between the different observed mechanism types. Specifically, we observe that lateral inhibition generates competition among neurons, homoeostasis helps to give each neuron a fair chance to compete, and that in such a setup excitatory learning leads to learning prototypical inputs as receptive fields (largely independent of the learning rule used).

Deep Artificial Neural Network (ANN) architectures such as GoogLeNet (Szegedy et al., 2015) and VGG-16 (Simonyan and Zisserman, 2014) have successfully pushed the state-of-the-art classification error rates to new levels on challenging computer vision benchmarks like ImageNet (Russakovsky et al., 2015). Inference in such very large networks, i.e., classification of an ImageNet frame, requires substantial computational and energy costs, thus limiting their use in mobile and embedded applications.

Recent work have shown that the event-based mode of operation in SNNs is particularly attractive for reducing the latency and computational load of deep neural networks (Farabet et al., 2012; O'Connor et al., 2013; Neil et al., 2016; Zambrano and Bohte, 2016). Deep SNNs can be queried for results already after the first output spike is produced, unlike ANNs where the result is available only after all layers have been completely processed (Diehl et al., 2015). SNNs are also naturally suited to process input from event-based sensors (Posch et al., 2014; Liu et al., 2015), but even in classical frame-based machine vision applications such as object recognition or detection, they have been shown to be accurate, fast, and efficient, in particular when implemented on neuromorphic hardware platforms (Neil and Liu, 2014; Stromatias et al., 2015; Esser et al., 2016). SNNs could thus play an important role in supporting, or in some cases replacing deep ANNs in tasks where fast and efficient classification in real-time is crucial, such as detection of objects in larger and moving scenes, tracking tasks, or activity recognition (Hu et al., 2016).

Multi-layered spiking networks have been implemented on digital commodity platforms such as FPGAs (Neil and Liu, 2014; Gokhale et al., 2014), but spiking networks with more than tens of thousands of neurons can be implemented on large-scale neuromorphic spiking platforms such as TrueNorth (Benjamin et al., 2014; Merolla et al., 2014) and SpiNNaker (Furber et al., 2014). Recent demonstrations with TrueNorth (Esser et al., 2016) show that CNNs of over a million neurons can be implemented on a set of chips with a power dissipation of only a few hundred mW. Given the recent successes of deep networks, it would be advantageous if spiking forms of deep ANN architectures such as VGG-16 can be implemented on these power-efficient platforms while still producing good error rates. This would allow the deployment of deep spiking networks in combination with an event-based sensor for real-world applications (Orchard et al., 2015a; Serrano-Gotarredona et al., 2015; Kiselev et al., 2016).

In order to bridge the gap between Deep Learning continuous-valued networks and neuromorphic spiking networks, it is necessary to develop methods that yield deep *Spiking* Neural Networks (SNNs) with equivalent error rates as their continuous-valued counterparts. Successful approaches include direct training of SNNs using backpropagation (Lee et al., 2016), the SNN classifier layers using stochastic gradient descent (Stromatias et al., 2017), or modifying the transfer function of the ANNs during training so that the network parameters can be mapped better to the SNN (O'Connor et al., 2013; Esser et al., 2015; Hunsberger and Eliasmith, 2016). The largest architecture trained by Hunsberger and Eliasmith (2016) in this way is based on AlexNet (Krizhevsky et al., 2012). While the results are promising, these novel methods have yet to mature to the state where training spiking architectures of the size of VGG-16 becomes possible, and the same state-of-the-art error rate as the equivalent ANN is achieved.

A more straightforward approach is to take the parameters of a pre-trained ANN and to map them to an equivalent-accurate SNN. Early studies on ANN-to-SNN conversion began with the work of Perez-Carrasco et al. (2013), where CNN units were translated into biologically inspired spiking units with leaks and refractory periods, aiming for processing inputs from event-based sensors. Cao et al. (2015) suggested a close link between the transfer function of a spiking neuron, i.e., the relation between input current and output firing frequency to the activation of a rectified linear unit (ReLU), which is nowadays the standard model for the neurons in ANNs. They report good performance error rates on conventional computer vision benchmarks, converting a class of CNNs that was restricted to having zero bias and only average-pooling layers. Their method was improved by Diehl et al. (2015), who achieved nearly loss-less conversion of ANNs for the MNIST (LeCun et al., 1998) classification task by using a *weight normalization* scheme. This technique rescales the weights to avoid approximation errors in SNNs due to either excessive or too little firing of the neurons. Hunsberger and Eliasmith (2016) introduced a conversion method where noise injection during training improves the robustness to approximation errors of the SNN with more realistic biological neuron models. Esser et al. (2016) demonstrated an approach that optimized CNNs for the TrueNorth platform which has binary weights and restricted connectivity. Zambrano and Bohte (2016) have developed a conversion method using spiking neurons that adapt their firing threshold to reduce the number of spikes needed to encode information.

Analysis of the layout - Main Body

Title
Abstract

Introduction
Related Work*

The Dataset
The Architecture
Reducing Overfitting → Main Body
Details of Learning
Results

Discussion

References

Congrats, after the introduction, your readers have well realized that you have done enough work for this paper, and your method/result seems great. Now, tell them more details!

1. Describe your method/Architecture
2. Describe mathematical formulas
3. Describe the experiments you have done.
    a. Datasets
    b. Figures
    c. Figure Notes
    d. Conclusion
4. Describe the tricks and skills
5. ***Do Not*** use any formulas, figures or tables without description or notes

Analysis of the layout - Main Body

Spiking Neural Networks

- Leaky Integrate-and-Fire Neuron
- Winner-Take-All Circuit

Using Backpropagation in SNNs

- Transfer functions and derivatives
- Initialization and Error normalization

Regularization

- Weight Regularization
- Threshold Regularization

Results and Discussion

$$a_i \approx \frac{s_i}{\gamma V_{th,i}} + \frac{\sigma \sum_{j=1,j\neq i}^{n} \kappa_{ij} a_j}{\gamma}, \text{ where } s_i = \sum_{k=1}^{m} w_{ik} x_k. \tag{5}$$

Refractory periods are not considered here since the activity of neurons in SNNs is rarely dominated by refractory periods in a normal operating regime. For example, we used a refractory period of 1 ms and the event rates of individual neurons were kept within a few tens of events per second (eps). Eq. (5) is consistent with (4.9) in [6] without WTA terms. It can also be simplified to a spiking version of a rectified-linear unit by introducing a unit threshold and non-leaky membrane potential as in [23]. Directly differentiating (5) yields the backpropagation equations

$$\frac{\partial a_i}{\partial s_i} \approx \frac{1}{\gamma V_{th,i}}, \frac{\partial a_i}{\partial w_{ik}} \approx \frac{\partial a_i}{\partial s_i} x_k, \frac{\partial a_i}{\partial V_{th,i}} \approx \frac{\partial a_i}{\partial s_i}\Big(-\gamma a_i + \sigma \sum_{j\neq i}^{n} \kappa_{ij} a_j\Big), \frac{\partial a_i}{\partial \kappa_{ih}} \approx \frac{\partial a_i}{\partial s_i}(\sigma V_{th,i} a_h), \tag{6}$$

$$\begin{bmatrix} \frac{\partial a_1}{\partial x_k} \\ \vdots \\ \frac{\partial a_n}{\partial x_k} \end{bmatrix} \approx \frac{1}{\sigma} \begin{bmatrix} q & \cdots & -\kappa_{1n} \\ \vdots & \ddots & \vdots \\ -\kappa_{n1} & \cdots & q \end{bmatrix}^{-1} \begin{bmatrix} \frac{w_{1k}}{V_{th,1}} \\ \vdots \\ \frac{w_{nk}}{V_{th,n}} \end{bmatrix} \tag{7}$$

where $q = \gamma/\sigma$. When all the lateral inhibitory connections have the same strength ($\kappa_{ij} = \mu, \forall i, j$) and are not learned, $\partial a_i / \partial \kappa_{ih}$ is not necessary and (7) can be simplified to

$$\frac{\partial a_i}{\partial x_k} \approx \frac{\partial a_i}{\partial s_i} \frac{\gamma}{(\gamma - \mu\sigma)} \left(w_{ik} - \frac{\mu\sigma V_{th,i}}{\gamma + \mu\sigma(n-1)} \sum_{j=1}^{n} \frac{w_{jk}}{V_{th,j}} \right). \tag{8}$$

We consider only the first-order effect of the lateral connections in the derivation of gradients. Higher-order terms propagating back through multiple lateral connections are neglected for simplicity. This is mainly

Analysis of the layout - Main Body

Datasets:

The PI MNIST task was used for performance evaluation [15]. MNIST is a hand written digit classification dataset consisting of 60,000 training samples and 10,000 test samples. The permutation-invariant version was chosen to directly measure the power of the fully-connected classifier. By randomly permuting the input stimuli

Results:

Table 2: Comparison of accuracy of different models on PI MNIST without unsupervised pre-training or cost function (except SNN([22]) and SNN([11])) and N-MNIST [24].

Network	# units in HLs	Test accuracy (%)
ANN ([27], Drop-out)	4096-4096	98.99
ANN ([29], Drop-connect)	800-800	98.8
ANN ([8], maxout)	240 × 5-240 × 5	99.06
SNN ([22])^{a,b}	500-500	94.09
SNN ([11])^{a}	500-300	98.6
SNN ([4])	1200-1200	98.64
SNN ([23])	200-200	97.8
SNN (SGD, This work)	800	[98.56, 98.64, 98.71]*
SNN (SGD, This work)	500-500	[98.63, 98.70, 98.76]*
SNN (ADAM, This work)	300-300	[98.71, 98.77, 98.88]*
N-MNIST (centered), ANN ([21])	CNN	98.3
N-MNIST (centered), SNN ([21])	CNN	95.72
N-MNIST (uncentered), SNN (This work)	500	[98.45, 98.53, 98.61]*

a: pretraining, b: data augmentation, *:[min, average, max] values over epochs [181, 200].

...(on-event for intensity increase, off-event for intensity decrease), we separated events into two channels based on the event type. Table 2 shows that our result of 98.53% with 500 hidden units is the best N-MNIST result with SNNs reported to date...

Analysis of the layout - Main Body

3.2 Winner-Take-All (WTA) Circuit

We found that the accuracy of SNNs could be improved by introducing a competitive recurrent architecture called WTA circuit in certain layers. In a WTA circuit, multiple neurons form a group with lateral inhibitory connections. Thus, as soon as any neuron produces an output spike, it inhibits all other neurons in the circuit and prevents them from spiking [25]. In this work, all lateral connections in a WTA circuit have the same strength, which reduces memory and computational costs for implementing them. The amount of lateral inhibition applied to the membrane potential is designed to be proportional to the inhibited neuron's membrane potential threshold (see (4) in Section 4.1). With this scheme, lateral connections inhibit neurons having small V_{th} weakly and those having large V_{th} strongly. This improves the balance of activities among neurons during training. As shown in Results, WTA competition in the SNN led to remarkable improvements, especially in networks with a single hidden layer. The WTA circuit also improves the stability and speed of training.

Lead In

What it is?

How it is in your work

What's the result?

Analysis of the layout - Main Body

2.2.6. Spiking Max-Pooling Layers

Most successful ANNs use max-pooling to spatially down-sample feature maps. However, this has not been used in SNNs because computing maxima with spiking neurons is non-trivial. Instead, simple average pooling used in Cao et al. (2015), Diehl et al. (2015), results in weaker ANNs being trained before conversion. Lateral inhibition, as suggested in Cao et al. (2015), does not fulfill the job properly, because it only selects the winner, but not the actual maximum firing rate. Another suggestion is to use a temporal Winner-Take-All based on time-to-first-spike encoding, in which the first neuron to fire is considered the maximally firing one (Masquelier and Thorpe, 2007; Orchard et al., 2015b). Here we propose a simple mechanism for spiking max-pooling, in which output units contain gating functions that only let spikes from the maximally firing neuron pass, while discarding spikes from other neurons. The gating function is controlled by computing estimates of the pre-synaptic firing rates, e.g., by computing an online or exponentially weighted average of these rates. In practice we found several methods to work well, but demonstrate only results using

Analysis of the layout - End

Title
Abstract

Introduction
Related Work*

The Dataset
The Architecture
Reducing Overfitting
Details of Learning
Results

Discussion → End of the story

References

Congrats! The readers have finally and fully understand what you are doing in your paper. Is there any chance that the readers can involve your research?

- Reclaim your advantages and applications.
- Indicate what research field will benefit from your research.
- Describe what can be done in the future.
- As important (or even more important) as any other part.

Analysis of the layout - End

We have shown that our novel spike-based backpropagation technique for deep SNNs works both on standard benchmarks such as PI MNIST, but also on N-MNIST, which contains rich spatio-temporal structure in the events generated by a neuromorphic vision sensor. We improve the previous state-of-the-art of SNNs on both tasks and achieve accuracy levels that match those of conventional deep networks. Closing this gap makes deep SNNs attractive for tasks with highly redundant information or energy constrained applications, due to the benefits of event-based computation, and advantages of efficient neuromorphic processors [19]. We expect that the proposed technique can precisely capture the statistics of spike signals generated from event-based sensors, which is an important advantage over previous SNN training methods. Future work will extend our training approach to new architectures, such as CNNs and recurrent networks.

Analysis of the layout - End

This work presents two new developments. The first is a novel theory…

In addition to the improved SNN results on MNIST and CIFAR-10, this work presents for the first time, a spiking network implementation of VGG-16 and Inception-V3 models…

With BinaryNet (an 8-layer CNN with binary weights and activations tested on CIFAR-10) (Courbariaux et al., 2016), we demonstrated that low-precision models are well suited for conversion to spiking networks…

We are currently investigating spike encoding schemes that make more efficient use of temporal structure than the present rate-based encoding.

Mostafa et al. (2017) present such an approach where the precise spike time is used to train a network to classify MNIST digits with a single spike per neuron. Such a sparse temporal code clearly reduces the cost of repeated weight fetches which dominates in rate-encoded SNNs.

Finally, this conversion framework allows the deployment of state-of-the-art pre-trained high-performing ANN models onto energy-efficient real-time neuromorphic spiking hardware such as TrueNorth (Benjamin et al., 2014; Merolla et al., 2014; Pedroni et al., 2016).

Analysis of the layout - End

Our results show that a large, deep convolutional neural network is capable of achieving recordbreaking results on a highly challenging dataset using purely supervised learning. It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results. To simplify our experiments, we did not use any unsupervised pre-training even though we expect that it will help, especially if we obtain enough computational power to significantly increase the size of the network without obtaining a corresponding increase in the amount of labeled data. Thus far, our results have improved as we have made our network larger and trained it longer but we still have many orders of magnitude to go in order to match the infero-temporal pathway of the human visual system. Ultimately we would like to use very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images.

Useful Materials for future study

- Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification
- Unsupervised learning of digit recognition using spike-timing-dependent plasticity
- Training Deep Spiking Neural Networks using Backpropagation
- ImageNet Classification with Deep Convolutional Neural Networks
- [Paper Collection] https://github.com/amusi/daily-paper-computer-vision
- [Computer Vision Resources] https://github.com/jbhuang0604/awesome-computer-vision
- [Paper Search & Reader] https://arxiv.autoai.org/
- [Introduction Video by Feifei Li] Link

Thanks!

This slides can be downloaded at https://xzyaoi.github.io/teaching/