READING CHECKS WITH MULTILAYER GRAPH TRANSFORMER NETWORKS

Yann Le Cun *Léon Bottou* *Yoshua Bengio*

Speech and Image Processing Services Research Lab
AT&T Labs, 101 Crawfords Corner Road, Holmdel, NJ 07733, USA
yann@research.att.com

ABSTRACT

We propose a new machine learning paradigm called Multilayer Graph Transformer Network that extends the applicability of gradient-based learning algorithms to systems composed of modules that take graphs as input and produce graphs as output. A complete check reading system based on this concept is described. The system combines convolutional neural network character recognizers with graph-based stochastic models trained cooperatively at the document level. It is deployed commercially and reads million of business and personal checks per month with record accuracy.

1. INTRODUCTION

The most common technique for building document processing systems is to partition the task into manageable subtasks, such as field detection, word segmentation, or character recognition, and to build a separate module for each one. Typically, each module is trained, or manually optimized, outside of its context. After the complete system is assembled, a subset of the parameters of the modules is manually adjusted to maximize the overall performance. This last step is extremely tedious, time-consuming, and often suboptimal.

The recent history of Automatic Speech Recognition is here to remind us that training a recognizer by optimizing a global criterion (at the word or sentence level) is much preferable to merely training it on hand-segmented phonemes or other units. Several recent works have shown that the same is true for handwriting recognition [2, 5]: optimizing a word-level criterion is preferable to solely training a recognizer on pre-segmented characters because the recognizer can learn not only to recognize individual characters, but also to reject mis-segmented characters thereby minimizing the overall word error.

Let us consider a document analysis system whose performance on a given data set is measured by an objective function E (for example, a "soft" version of the classification error) that depends upon a large vector of tunable parameters W. The goal of training is to find the parameter vector that minimizes $E(W)$. Common wisdom suggests that this problem is intractable for most realistic cases since the minimization might be essentially combinatorial. However, if $E(W)$ can be made differentiable with respect to W, we can find its minimum using simple *gradient-based* techniques such as stochastic gradient descent or conjugate gradient. To ensure that $E(W)$ is differentiable we can build the overall system as a feed-forward network of differentiable modules. The function implemented by each module must be differentiable *almost everywhere* with respect the internal parameters of the module, and with respect to the module's inputs. If this is the case, a simple generalization of the well-known gradient back-propagation procedure, widely used

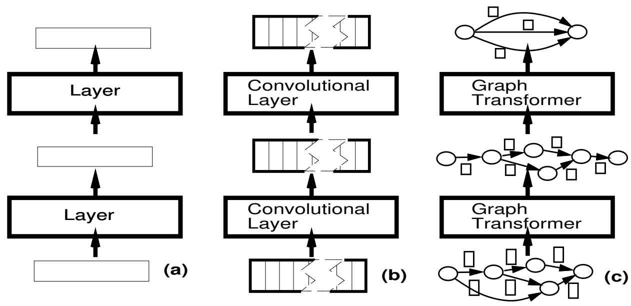

Figure 1. *(a) Traditional Neural Networks communicate fixed-size vectors between layer. (b) Convolutional Neural Networks and Recurrent Neural Networks can handle variable-length sequences of vectors. (c) Multilayer Graph Transformer Networks are composed of trainable modules that operate on and produce graphs whose arcs carry numerical information.*

for multilayer neural net training, can be used to efficiently compute the gradients of the objective function with respect to all the parameters in the system [3]. For example, let us consider a system built as a cascade of modules, each of which implements a function $X_n = F_n(W_n, X_{n-1})$, where X_n is a vector representing the output of the module, W_n is the vector of tunable parameters in the module (a subset of W), and X_{n-1} is the module's input vector (as well as the previous module's output vector). If we assume that the partial derivative of E with respect to X_n is known, the partial derivatives of E with respect to W_n and X_{n-1} can be computed using

$$\partial E/\partial W_n = \partial F/\partial W(W_n, X_{n-1})\partial E/\partial X_n$$

$$\partial E/\partial X_{n-1} = \partial F/\partial X(W_n, X_{n-1})\partial E/\partial X_n$$

where $\partial F/\partial W(W_n, X_{n-1})$ is the Jacobian of F with respect to W evaluated at the point (W_n, X_{n-1}), and $\partial F/\partial X(W_n, X_{n-1})$ is the Jacobian of F with respect to X. Using the above equation, we can compute the complete gradient of $E(W)$ working our way backwards from the output to the input as in the traditional back-propagation procedure.

2. GRADIENT-BASED LEARNING IN LARGE HETEROGENEOUS SYSTEMS

Traditional multilayer neural networks can be viewed as cascades of trainable modules (the layers) that communicate their states and gradients in the form of fixed-size vectors. Such representations lack flexibility for many applications that deal with variable length inputs, notably speech and handwriting recognition. Convolutional network architectures such as Time Delay Neural Networks (TDNN), Space

Displacement Neural Networks (SDNN) [2, 5], or recurrent networks, have been proposed to deal with variable-length sequences of vectors, and have been applied with success to optical character recognition, on-line handwriting recognition, spoken word recognition, and time-series prediction. However, these architectures still lack flexibility for tasks in which the state used to encode probability distributions over sequences of vectors (e.g. stochastic grammars). In such cases, the data is best represented using a *directed graph* in which each arc contains a vector. Each path in the graph represents a different sequence of vectors. Distributions over sequences can be represented by interpreting parts of the data associated with each arc as a penalty or likelihood.

One of the main points of the paper is to show that the gradient-based training procedure described in the previous section is not limited to networks simple modules that communicate through fixed-size vectors, but can be generalized to networks of modules called *graph transformers* that communicate their states and gradients in the form of directed graphs whose arcs carry numerical information (scalars or vectors). Graph transformers take one or more graphs as input and constructs a graph on its output (see figure 1). A back-propagation phase takes gradients with respect to the numerical information in the output graph, and computes gradients with respect to the numerical information attached to the input graphs, and with respect to the module's internal parameters. Gradient-based learning can be performed as long as differentiable functions are used to produce the numerical data in the output graph from the numerical data in the input graph, and from the functions parameters. We show that complete, trainable, document processing systems can be built within that framework. This is demonstrated with a practical system for reading bank checks. The core of the system is a universal character recognition module based on Convolutional Neural Networks. The check reading system is commercially deployed, reading several million handwritten and machine printed checks per month with record accuracy.

3. GRAPH TRANSFORMER NETWORKS

To help make the concept of graph transformer network concrete, we will describe an oversimplified example of trainable system built from three graph transformers in the context of handwriting recognition. The task of the system is to find the best segmentation of a handwritten word into characters (see figure 2). A word image is first cut into "pieces of ink" using heuristic image processing techniques (such as connected components, vertical projections,...). Each piece of ink may be a whole character or a piece of character. The bottom of figure 2 shows an example of a three-character word cut into four pieces of ink. A so-called *segmentation graph* G_{seg} is built to represent all the possible groupings of pieces of ink into characters. Each arc is associated with one piece of ink or with a combination of successive pieces (called a segment), such that each path in the graph goes once and only once through each piece of ink. To each path corresponds a different grouping of the pieces of ink into characters. A first graph transformer T_{rec}, which we will call the recognition transformer, creates its output graph G_{rec} by replicating the segmentation graph, replacing the segments by a positive penalty term that indicates how good a character the segment is. If the segment is a good-looking character, the penalty is small (close to 0), if it is an incomplete or bad-looking character, the penalty is larger. These numbers can be seen as negative log-likelihoods, or distances. They are generated from the segment images through a character scorer function R_w, parameterized by a vector w (e.g. a neural network with weight vector w). The cumulated penalty over a path in G_{rec} is a measure of badness of the corresponding segmentation. A second trans-

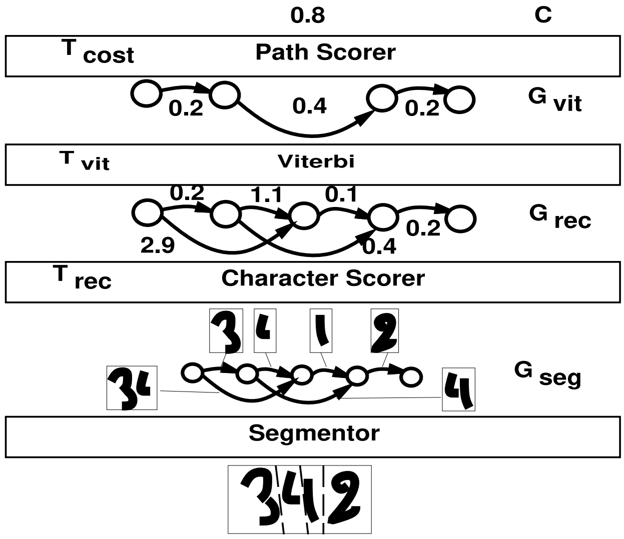

Figure 2. *A simple example of graph transformer machine that finds the best segmentation of a handwritten word.*

former T_{vit}, called a Viterbi transformer, takes this graph as input and produces a trivial graph G_{vit} whose only path is the lowest-penalty path in the segmentation graph. A third transformer T_{cost} takes G_{vit} and outputs a single number: the cumulated penalty C of G_{vit}. These three transformers are very special cases. T_{rec} changes the content of the arcs but not the structure of the graph. T_{vit} changes the graph structure, but it simply duplicates a subset of the arcs on its output without changing their content. T_{cost} outputs a single number. More complex transformers will be described in the main part of the paper.

3.1. Backpropagation in Graph Transformer Networks

Backpropagating gradients through the above system is very simple. We assume that the system is part of larger system whose training minimizes an objective function E. For each variable x used the forward pass (arc penalties, character scorer parameters....), the backpropagation phase will compute a corresponding partial derivative $\partial E/\partial x$. We assume that $\partial E/\partial C$, the gradient of E with respect to C, the cumulated penalty over the best path, is known. Since C is simply the sum of the penalties of each arc in G_{vit}, the gradients of E with respect to these penalties are all equal to $\partial E/\partial C$. Since G_{vit} is a subset of G_{rec}, the derivatives of E with respect to the penalties in G_{rec} are equal to $\partial E/\partial C$ for those arcs that appear in G_{vit}, and 0 for all others. The gradient of E with respect to w, the parameter vector of the character scorer R_w in T_{rec}, is simply the sum over all arcs in T_{rec} of $\partial E/\partial C \partial R/\partial w$. We have done nothing more than apply the chain rule, and the resulting algorithm is nothing more than a gradient-based version of the well-known Viterbi training procedure. We can view the above process as backpropagating gradients through an architecture whose dataflow graph changes with the input data. This is in contrast with traditional neural nets whose architecture is fixed. Now that we have explained the concept in a familiar situation, we will generalize the idea, and apply it to a practical case.

3.2. Previous Work

Numerous authors in speech recognition have used gradient-based training methods that integrate graph-based statistical models (notably HMM) with acoustic recognition modules, mainly gaussian mixture models, but also neural networks [4]. Similar ideas have been applied to handwriting recognition [2]. However, there has been no proposal for a systematic approach to multilayer graph-based trainable systems. The idea of transforming graphs into other graphs is at the basis of the concept of *weighted finite-state transducers* [7], but little effort has been devoted to building globally trainable systems out of transducers. The concept of building trainable systems by assembling modules and propagating gradients through them has been proposed for quite some time [3], and used extensively in practical systems [2]. However, it was not suggested that complex data structures such as graphs could be systematically used as state variables between modules. A different approach to graph-based trainable systems, called Input-Output HMM, was proposed in [1].

4. READING CHECK AMOUNTS WITH A GRAPH TRANSFORMER NETWORK

4.1. A Graph Transformer Network

The idea of graph transformer networks was used to build a check amount reading system. We now describe the successive graph transformation modules that allow our system to read the check amount. Each Graph Transformer produces a graph whose paths encode and score the current hypotheses considered at this stage of the system. All the functions used throughout the system to compute arc penalties are differentiable and possibly contain trainable parameters.

The input to the system a trivial graph with a single arc that carries the image of the whole check.

The field location transformer T_{field} first performs classical image analysis (including connected components, ink density histograms, etc...) and heuristically extracts rectangular zones that may contain the check amount. T_{field} produces an output graph, called the *field graph* such that each candidate zone found is associated with one arc that links the start node to the end node. Each arc contains the image of the zone, and a penalty term computed from simple features extracted from the zone (absolute position, size, aspect ratio,...).

The segmentation transformer T_{seg}, is similar to the one described in section 3.. It examines each zone contained in the field graph, generates a segmentation graph from it, setting preliminary arc penalties using geometrical heuristics. The preliminary penalty for a particular segmentation of a particular field is the sum of the arc penalties along the corresponding path.

The recognition transformer T_{rec} iterates over all segment arcs in the segmentation graph and runs a character recognizer on the corresponding segment image. In our case, the recognizer is a Convolutional Neural Network [6] described in more detail in section 6., whose weights constitute the largest and most important set of tunable parameters. The recognizer classifies segment images into one of 95 classes (full printable ASCII set) plus a "junk" class for unknown symbols or badly-formed characters. Each arc in the input graph is replaced by 96 arcs in the output graph. Each of those 96 arcs contains the label of one the classes, and a penalty that is the sum of the penalty of the corresponding arc in the input graph and the penalty associated with classifying the image in the corresponding class, as computed by the recognizer. Each path in this graph represents a possible character string for a field. This sequence of characters may or may not be a valid check amount.

The grammar transformer T_{gram} selects the paths of the recognition graph that represent valid character sequences for check amounts. This transformer takes two graphs as input: the recognition graph, and the grammar graph that encodes all possible well-formed amounts. The output of the grammar transformer, called the interpretation graph, contains all the paths in the recognition graph that are compatible with the grammar. The operation that combines the two input graphs to produce the output is called a *graph composition* (see [7] for a formal definition). To generate the interpretation graph we place a token on the start node of the recognition graph and a token on the start node of the grammar graph. We can simultaneously advance the two tokens along two arcs if the numerical information on the two arcs *match* according to a matching function. In our case, the matching function simply checks that the class labels on the two arcs are identical. When the two tokens are moved, an arc in the output graph is created. The output arc receives the class label of the two arcs, and a penalty that is simply the sum of the penalties of the originating arcs. **The Viterbi transformer** finally selects the path with the lowest accumulated penalty, corresponding to the best grammatically correct interpretations.

If a probabilistic score is desired, we can obtain one by computing the ratio between (a) the negative exponential of the total penalty of best path, and (b) the sum of the negative exponentials of the penalties of all the paths. The denominator is easily computed using the classical forward algorithm. We can directly compute its logarithm by proceeding forward in the graph and setting the penalty of each node to the *logsum* of the penalties of the incoming arcs added with the penalties of the upstream nodes. The penalty of the end node can be interpreted as the negative log-likelihood that the check contains a grammatically correct amount.

5. GRADIENT-BASED LEARNING

Each stage of this check reading system contains tunable parameters. While some of these parameters could be manually adjusted, for example the parameters of the field locator and segmentor, the vast majority of them *must* be learned, particularly the weights of the neural-net recognizer. Prior to globally optimizing the system, the parameters of the the field locator and the segmentor are initialized by hand, while the parameters of the neural net character recognizer are initially trained on a database of pre-segmented and labeled characters. Then, the entire system is trained globally from whole check images labeled with the correct amount. No explicit segmentation of the amounts is needed to train the system: it is trained at the check level.

The objective function E minimized by our global training procedure is a discriminant criterion similar to the Maximum Mutual Information criterion sometimes used in speech recognition systems. This criterion is the difference between the accumulated penalty of the *correct answer*, and the negative log-likelihood for the full grammar, as computed by the forward algorithm described above. For the training phase, we therefore pass the interpretation graph through two grammar transformers, one with a grammar that only contains the correct answer, and one with the normal grammar, that output the corresponding negative log-likelihoods, the difference of which is E. The partial derivatives of E with respect to the arc penalties in the interpretation graph are simply computed by backpropagation. Since all the penalties throughout the system are computed with differentiable function from the penalties or numerical data at previous layers, we can easily backpropagate partial derivatives of E with respect to all the penalties and all the parameters. In doing so, we do nothing more than apply the chain rule, albeit to a somewhat complicated function.

6. SHAPE RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

The recognizer used in the Check Reading System is a convolutional neural network coined LeNet5. Convolutional neural nets are specifically designed to recognize 2D shapes with a high degree of invariance with respect to translations,

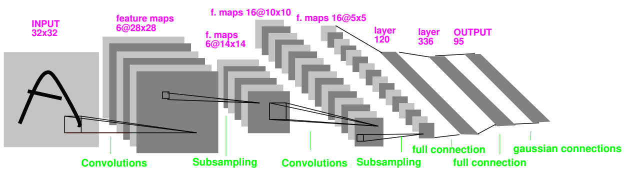

Figure 3. *Architecture of LeNet 5. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.*

scaling, skewing, and other distortions (see [5] for a review). They can directly accept images with no other preprocessing than approximate size normalization and centering. They have had numerous applications particularly in handwriting recognition. The architecture of LeNet5 is shown in figure 3. In a convolutional net, each unit takes its input from a local "receptive field" in the previous layer, forcing it to extract a local feature. Units located at different places on the image are grouped in planes, called *feature maps*, within which units are constrained to share a single set of weights. This makes the operation performed by a feature map shift invariant, and equivalent to a convolution followed by squashing functions. This *weight-sharing* technique greatly reduces the number of free parameters, thereby minimizing the necessary amount of training samples. A layer is composed of multiple feature maps, that share different sets of weights, and extract different types of features.

Complete networks are composed of multiple convolutional layers, extracting features of increasing complexity and abstraction. Reduced sensitivity to shifts and distortions is built into the system by inserting subsampling layers between the convolution layers. This forces the higher layers to extract progressively more global, and less position-sensitive features. It is important to stress again that *all* the weights in such a network are trained by gradient descent, none of the extracted features are hand-designed. The training process causes convolutional networks to automatically synthesize relevant features. Computing the gradient is done with a slightly modified version of the classical backpropagation procedure. LeNet5 has 401,000 connections, but only about 90,000 free parameters because of the weight sharing.

7. RESULTS

A version of the above system was implemented and tested on a representative mixture of business checks (mostly machine printed) and personal checks (mostly handwritten). The neural network classifier was initally trained on 500,000 images of character images from various origins spanning the entire printable ASCII set. This contained both handwritten and machine-printed characters that had been previously size normalized at the string level. Additional images were generated by randomly distorting the original images using simple affine transformations of the images. The network was then further trained on character images that had been automatically segmented from check images and manually truthed. The network was also trained to reject non-characters that resulted from segmentation errors. The recognizer was then inserted in the check reading system and trained globally (at the field level) on check images.

The performance of the system at the check level is 1% error, 50% correct, and 49% rejected, which places it above the threshold of economic viability. The test set is different from the training set and composed of a "natural" mixture of business checks and personal checks. On business checks, which are generally machine-printed, the amount is relatively easy to read, but quite difficult to find due to the lack of standard for business check layout. On the other hand, the amount on personal checks is easy to find but much harder to read. The use of a grammar for check amounts seem particularly helpful for reading business checks.

Independent test by systems integrators have shown the superiority of this system over several commercially available systems. The system was integrated in NCR's latest line of back-office check reading machines. It was fielded in a bank on June 1996, and has been reading millions of checks per month since then.

8. CONCLUSION

We have presented a new architecture for trainable systems that significantly extends the domain of applications of gradient-based learning. We have shown that all the steps of a document analysis system can be formulated as graph transformers through which gradients can be back-propagated. Even in the non-trainable parts of the system, the design philosophy in terms of graph transformation provides a clear separation between domain-specific heuristics and generic, procedural knowledge (the graph transformation paradigm). A Check Reading System, based on these ideas was built and commercially deployed.

Although this paper presents a small number of examples of trainable graph transformer modules, it is clear that the concept can be applied to many situations where the domain knowledge or the state information can be represented by graphs. This is the case in many audio signal recognition tasks, and visual scene analysis applications. Future work will attempt to apply graph transformer networks to such problems, with the hope of allowing more reliance on automatic learning, and less on detailed engineering.

REFERENCES

[1] Y. Bengio and P. Frasconi. An input/output HMM architecture. In G. Tesauro, D Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 427–434. MIT Press, Cambridge, MA, 1996.

[2] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(5), 1995.

[3] L. Bottou and P. Gallinari. A framework for the cooperation of learning algorithms. In D. Touretzky and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 3, Denver, 1991. Morgan Kaufmann.

[4] P. Haffner, M. Franzini, and A. Waibel. Integrating time-alignment and neural networks for high performance continuous speech recognition. In *Proc. of ICASSP 91*. IEEE, 1991.

[5] Y. Le Cun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[6] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Back-propagation applied to handwritten zipcode recognition. *Neural Computation*, 1(4), January 1990.

[7] F. Pereira, M. Riley, and R. Sproat. Weighted rational transductions and their application to human language processing. In *ARPA Natural Language Processing workshop*, 1994.

READING CHECKS WITH MULTILAYER GRAPH TRANSFORMER NETWORKS

Yann Le Cun , Léon Bottou and Yoshua Bengio

Speech and Image Processing Services Research Lab
AT&T Labs, 101 Crawfords Corner Road, Holmdel, NJ 07733, USA
yann@research.att.com

We propose a new machine learning paradigm called Multilayer Graph Transformer Network that extends the applicability of gradient-based learning algorithms to systems composed of modules that take graphs as input and produce graphs as output. A complete check reading system based on this concept is described. The system combines convolutional neural network character recognizers with graph-based stochastic models trained cooperatively at the document level. It is deployed commercially and reads million of business and personal checks per month with record accuracy.