Global Training of Document Processing Systems using Graph Transformer Networks.

Category: Document image analysis
Keywords: Learning Algorithm, Graph Transformers

Abstract

We propose a new machine learning paradigm called Graph Transformer Networks that extends the applicability of gradient-based learning algorithms to systems composed of modules that take graphs as inputs and produce graphs as output. Training is performed by computing gradients of a global objective function with respect to all the parameters in the system using a kind of back-propagation procedure.

A complete check reading system based on these concept is described. The system uses convolutional neural network character recognizers, combined with global training techniques to provides record accuracy on business and personal checks. It is presently deployed commercially and reads million of checks a month.

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.

For example, a character recognition module can be trained to recognize well-formed individual characters. However, the role of the recognizer in the context of the entire system is usually quite different. Very often, character recognizers are expected, not only to recognize well-formed individual characters, but also to reject badly segmented characters and other non-characters. They are also expected to produce reliable scores that can be used by a post-processor, such as a statistical language model. For example, if the task is to read the amount on bank checks, the overall objective function that must be minimized is the percentage of checks that must be rejected in order to attain a given error rate on the accepted checks. Merely training the character recognizer module to minimize its classification error on individual characters will not minimize the global objective function. Ideally, we would like to find a good minimum of the global objective function with respect to all the parameters in the system. In most practical cases, this optimization problem appears intractable. However, if the objective function E measuring the performance measure can be made differentiable with respect to the systems tunable parameters W, we can find its minimum $\min_W E(W)$ using *gradient-based* techniques such as stochastic gradient descent or conjugate gradient.

A simple technique to ensure that the global objective function $E(W)$ is differentiable is to 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 (e.g. the weights of a Neural Net character recognizer in the case of a charcter recognition module), and with respect to the module's inputs. If this is the case, a simple generalization of the well-known back-propagation procedure (Rumelhart, Hinton and Williams, 1986), widely used 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 (Bottou and Gallinari, 1991). 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. The limited flexibility of

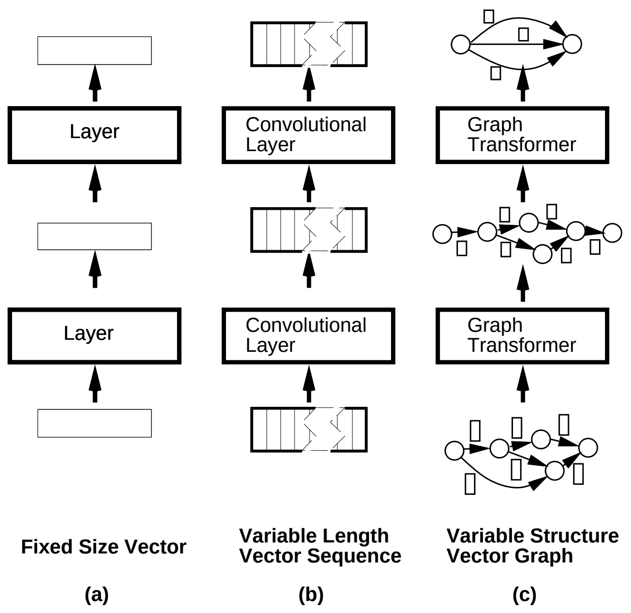

Figure 1: *(a) Traditional Neural Networks communicate fixed-size vectors between layer. (b) Convolutional Neural Networks and Recurrent Neural Networks can handle variable-length sequences of vectors. (c) Multilayer Graph Transformer Networks are composed of trainable modules that operate on and produce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

fixed-size vectors for data representation is a serious deficiency for many applications, notably for tasks that deal with variable length inputs (e.g speech and handwriting recognition), or for tasks that require encoding relationships between objects or features whose number and nature can vary (invariant perception, scene analysis, reasoning).

Convolutional network architectures such as Time Delay Neural Networks (TDNN), Space Displacement Neural Networks (SDNN) (Le Cun and Bengio, 1995), or recurrent network, have been proposed to deal with variable-length sequences of vectors. They have been applied with great 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 must encode more than simple vector sequences. A notable example is when the state is used to encodes probability distributions over sequences of vectors (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 score or likelihood. Distributions over sequences are particularly handy for modelling linguistic knowledge in speech or handwriting recognition systems: each sequence, i.e. each path in the graph, represents an alternative interpretation of the input. Successive processing modules progressively refine the interpretation. For example, a speech recognition system starts with a single sequence of acoustic vectors, transforms it into a lattice of phonemes (distribution over phoneme sequences), then into a lattice of words (distribution over word sequences), then into a single sequence of words representing the best interpretation.

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 network 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 its 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.

The second goal of the paper is to show that the functions implemented by many of the modules used in typical document processing systems (and other image recognition systems), though commonly thought to be combinatorial in nature, are indeed differentiable with respect to their internal parameters as well as with respect to their inputs, and are therefore usable as part of a globally trainable system.

Lastly, we demonstrate the use of globally-trained graph-transformer networks on a check reading task. The check reading system we describe here is commercially deployed, reading several million handwritten and machine printed checks per month with record accuracy.

3 Graph Transformer Networks

To help make the of graph transformer network more 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 number that indicates how good a character the segment is. If the segment is a good-looking character, the number is small (close to 0), if it is an incomplete or bad-looking character, the number is larger. These numbers can be seen as negative log-likelihoods, distances, or simply costs. 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 cost over a path in G_{rec} is the badness (or cost) of the corresponding segmentation. A second transformer T_{vit}, called a Viterbi transformer, takes this graph as input and produces a trivial graph G_{vit} whose only path is the lowest-cost path in the segmentation graph. A third transformer T_{cost} takes G_{vit} and outputs a single number: the cumulated cost 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 transformer 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 costs, 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 cost over the best path, is known. Since C is simply the sum of the costs of each arc in G_{vit}, the gradients of E with respect to these costs 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 costs 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. The difference with traditional neural networks, and other systems in which gradient-based learning is commonly applied, is that the architecture (or dataflow graph) through which we propagate states and backpropagate gradients changes with the input data.

3.2 Previous Work

Over the last few years, many authors have proposed ad-hoc ways of building trainable systems by combining graph-based models with other learning methods such as neural networks, notably for speech recognition (Bourlard and Wellekens, 1989; Haffner,

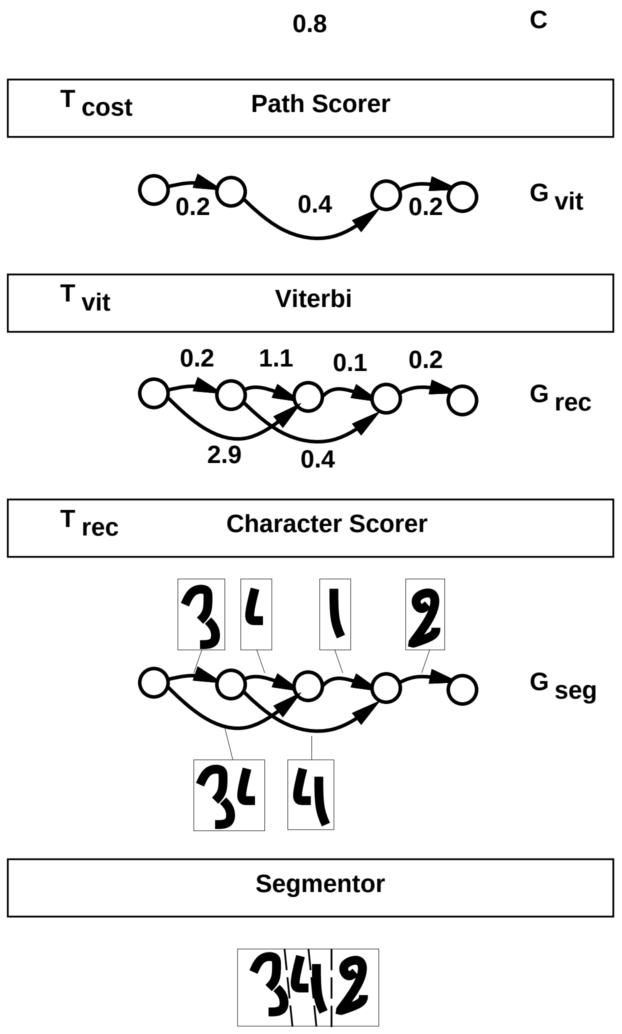

Figure 2: *A simple example of graph transformer machine that finds the best segmentation of a handwritten word. The word is segmented using heuristic methods, a segmentation graph is built, each segment is scored, and the best overall segmentation is extracted. Gradients can be easily backpropagated through the entire system.*

Franzini and Waibel, 1991), and handwriting recognition (see (Bengio et al., 1995a) for a review). However, there has been no proposal for a systematic approach to multilayer graph-based trainable systems. The idea of transforming graphs into other graphs has received considerable interest in computer science, through the concept of weighted finite-state transducers (Pereira, Riley and Sproat, 1994). Transducers have been applied to speech recognition and language translation, and proposals have been made for handwriting recognition (Guyon, Schenkel and Denker, 1996). This line of work has been mainly focused on the algebraic aspects of combining transducers and graphs (called acceptors in this context), but very little effort has been devoted to building globally trainable large system out of transducers. Learning has been the focus of another line of work, called Input-Output HMM, which uses graph-based models to transform sequences into distributions over other sequences (Bengio and Frasconi, 1996). However IOHMM do not use the idea of using dynamically created graphs as ways of coding distributions over sequences. The concept of building trainable systems by assembling modules and propagating gradients through them has been proposed for quite some time (Bottou and Gallinari, 1991), and used extensively in practical systems (Bengio et al., 1995a). However, it was not suggested that complex data structures such as graphs could be systematically used as state variables between modules.

4 Reading Check Amounts with a Graph Transformer Network

The idea of graph transformer networks was used to build a check amount reading system. The system must find candidate fields, select the field that is most likely to contain the amount, segment the fields into candidate characters, read an scores the candidate characters, and finally find the best interpretation of the amount using contextual knowledge, namely a stochastic grammar for check amounts. We describe now a cascade of graph transformers (cf. figure 3) that starts with a check image and performs all these operations.

4.1 A Graph Transformer Network

We now describe the successive graph transformations that allow our network to read the check amount. Each Graph Transformer produces a graph whose paths encode and score the current hypothesis considered at this stage of the system.

The input to the system a trivial graph whose only arc carries the image of the whole check (cf. figure 4).

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* (cf. figure 5) 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 ter-

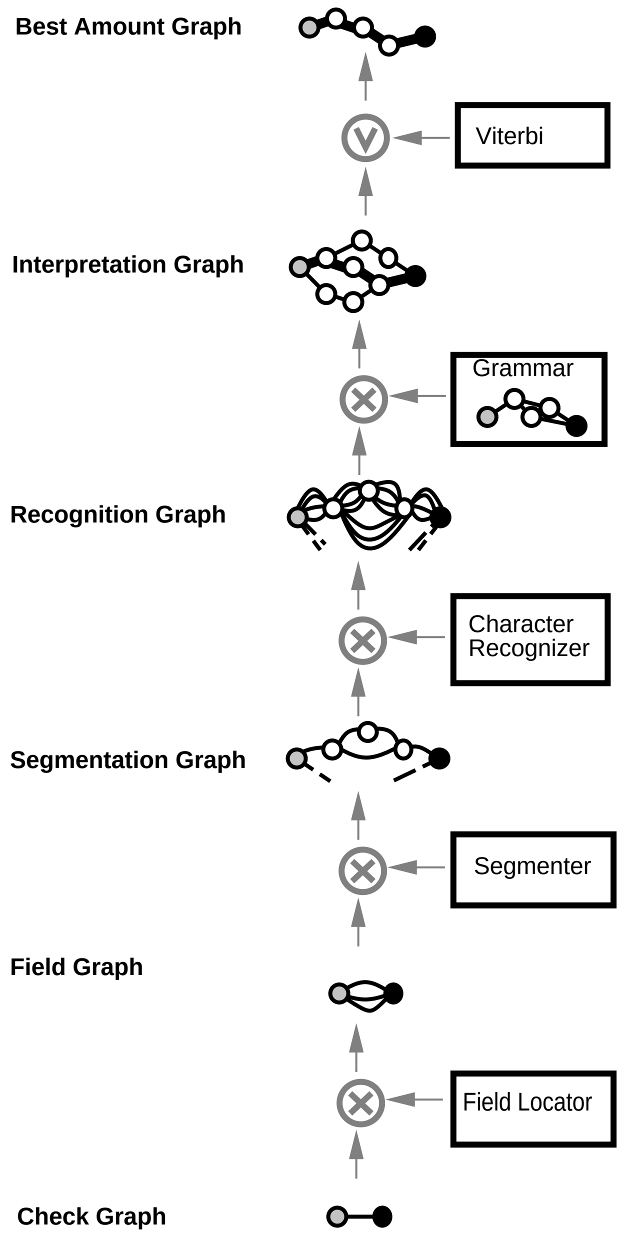

Figure 3: *A complete check amount reader implemented as a single cascade of graph transformer modules. Successive graph transformations progressively extract high level information.*

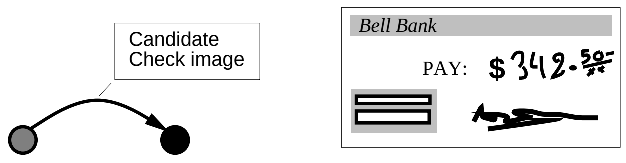

Figure 4: *The initial graph contains a single path representing the whole check image.*

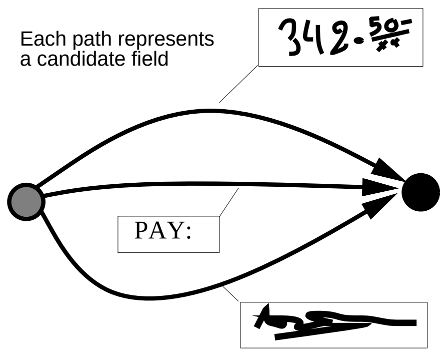

Figure 5: *The field graph represents the few rectangular areas that may contain the check amount. Each arc carries a rough score based on the features extracted by the initial image analysis.*

m" computed from simple features extracted from the zone (absolute position, size, aspect ratio,...). The penalty term is close to zero if the features suggest that the field is a likely candidate, and is large if the field is deeemed less likely to be an amount. If the penalty function is differentiable, its parameter are globally tunable.

The segmentation transformer T_{seg}, similar to the one described in section 3 examines each zone contained in the field graph, and cuts each image into "pieces of ink" using heuristic image processing techniques. Each piece of ink may be a whole character or a piece of character. Each arc in the field graph is replaced by its corresponding segmentation graph that represents all possible groupings of pieces of ink. Each field segmentation graph is appended to an arc that only contains the penalty of the field in the field graph. Each arc carries the segment image, together with a penalty that provides a first evaluation of the likelihood that the segment actually is character. This penalty is obtained with a differentiable function that combines a few simple features such as the space between the pieces of ink, or the compliance of the segment image with a global baseline, using a few tunable parameters. The segmentation graph represents *all* the possible segmentation of *all* the field images. We can compute the penalty for one segmented field by adding the arc penalties along the corresponding path. As before using a differentiable function for computing the penalties will ensure that the parameters can be optimized globally.

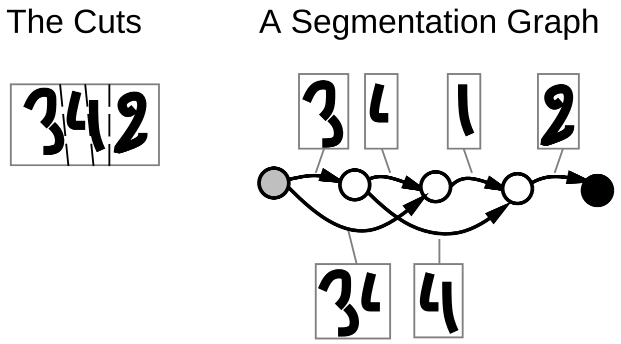

Figure 6: *The segmentation graph of a single where each path represents one possible grouping of the "pieces of ink" into a sequence of characters.*

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 (Le Cun et al., 1990a) whose weights constitute the largest and most important subset of tunable parameters. The recognizer classifies images into one of 15 classes (10 digits, asterisk, Dollar sign, dash, comma, period), plus "junk" classes for other symbols or badly-formed characters. Each arc in the input graph is replaced by 16 arcs in the output graph. Each of those 16 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. In other words, the recognition graph represents a weighted treillis of scored character classes. Each path in this graph represents a possible character string for the corresponding field. We can compute a penalty for this interpretation by adding the penalties along the path. 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. The grammar graph contains all possible sequences of symbols that constitute a well-formed amount. 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*. This algorithm was formally studied in the context of finite-state transducers, and although its general form is quite complex, (Pereira, Riley and Sproat, 1994), it can be explained using a simple metaphor. 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 the the numerical information attached with the two arcs *match* according to a matching function. In our case, the matching function simply check that the class labels on the two arcs are indentical. When the two tokens are moved, an arc in the output graph is created. A differentiable function is used to compute the data attached to the output arc from the data attached to the input arcs. In our case, the output arc receives the class label of the two arcs, and a penalty computed by simply summing the penalties of the two input arcs (the recognizer score, and the arc penalty in the grammar graph). The above procedure produces a tree, but simple techniques can be used to avoid generating multiple copies of certain subgraphs by detecting when a particular output state has already been seen. Each path in the interpretation graph represents one interpretation of one segmentation of one field on the check. The sum of the penalties along the path represents the "badness" of the corresponding interpretation and combines evidence from each modules along the process, as well as from the grammar.

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. Such path probabilities are clearly positive and sum to one. The denominator is easily computed using the forward algorithm widely used in Hidden Markov Model-based speech recognition systems (Rabiner, 1989). 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.

A similar procedure allows us to compute the negative log-likelihood of *any* amount we choose. For instance, if we compute ratio between (a) the sum of the negative exponential of the penalties of all paths representing the correct amount, and (b) the sum of negative exponential of the penalties of all paths, we will obtain the log-likelihood attributed by the system to the correct answer. This is particularly useful for global training as will be discussed below.

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 optimize the system, each module parameters must be initialized with reasonable values. The parameters of the the field locator and the segmentor can be initialized by hand. The parameters of the neural net character recongizer can be initialized by training on a database of pre-segmented and labelled characters. Then, the entire system can be trained globally from whole check images labelled 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 used in speech recognition systems based on Hidden Markov Models (Rabiner, 1989). 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. Since the grammar includes the correct answer, the objective function is always positive. It is zero if the penalties of the wrong answers are infinitely larger than the penalty of the correct answer. For the training phases, we therefore pass the interpretation graph through two transformers (a "constrained forward algorithm" and a "free forward algorithm") 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 back-propagation. All the arc penalties along the paths representing the correct answer will have a positive gradient, while all other arcs will have negative gradients.

Once we have those, we back-propagate gradients through the grammar transformer and obtain partial derivatives with respect to the penalties on the recognition graph and on the grammar graphs. This can be done if we keep track of which arcs in the grammar and recognition were used to generate which arc on the interpretation graph. Since these scores s_{int} are simple combinations of the arc scores of the grammar graph and of the arc scores of the recognition graph, we can compute the derivatives $\partial L/\partial w_{\text{gram}}$ with respect to the tunable parameters of the grammar and the derivatives $\partial L/\partial s_{\text{rec}}$ with respect to the arc scores of the recognition graph.

Since these scores s_{rec} are simple combination of the arc scores of the segmentation graph and of the scores returned by the isolated character recognizer, we can first compute the derivatives $\partial L/\partial s_{\text{net}}$ with respect to the outputs of the character recognizer and therefore compute the derivatives $\partial L/\partial w_{\text{net}}$ of all tunable parameters of the recognizer. We can also compute the derivatives $\partial L/\partial s_{\text{seg}}$ with respect to all scores of the segmentation network.

Since these scores s_{seg} are simple combination of the arc scores of the field graph and of the tunable parameters of the segmenter, we can compute the derivatives $\partial L/\partial w_{\text{seg}}$ with respect to the tunable parameters of the segmenter and the derivatives $\partial L/\partial s_{\text{field}}$ with respect to the arc scores of the field graph.

Finally we can use these latter derivatives to com-

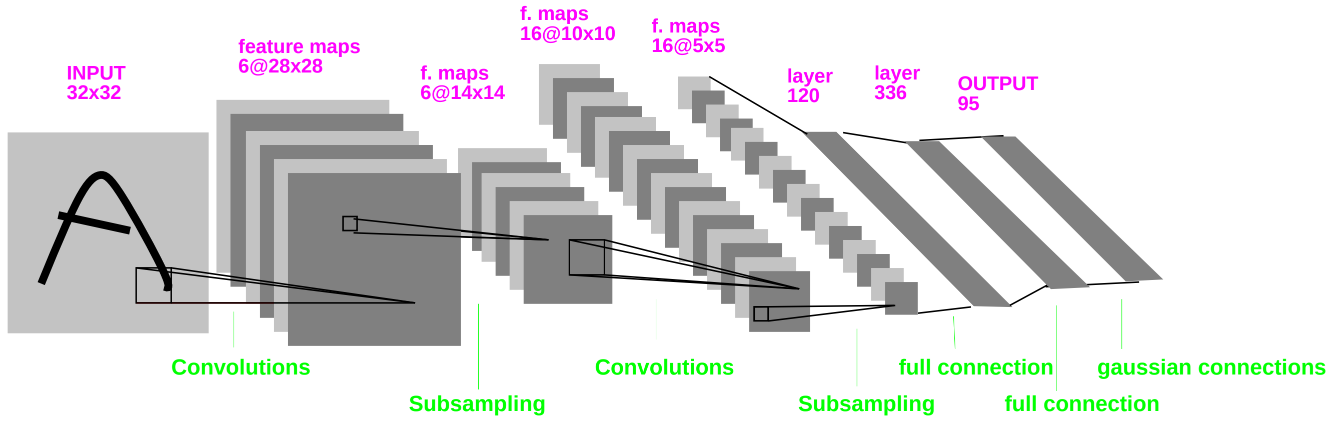

Figure 7: *Architecture of LeNet 5. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.*

pute the derivatives $\partial L / \partial w_{\text{field}}$ with respect to the parameters of the field locator.

In doing so, we have done nothing more than apply the chain rule, albeit in a very complicated architecture.

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, scaling, skewing, and other distortions. They can directly accept images was no preprocessing but a simple size normalization and centering. They have had numerous applications in handwriting recognition (Le Cun et al., 1990b; Bengio et al., 1995a; Bengio et al., 1995b; Le Cun et al.,), and object location in images, particulaly faces (Vaillant, Monrocq and Le Cun, 1994). The architecture of LeNet5 is shown in figure 7. In a convolutional net, each unit takes its input from a local "receptive field" on the layer below, forcing it to extract a local feature. Furthermore, 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. A single layer is formed of multiple feature maps, extracting different features types.

Complete networks are formed of multiple convolutional layers, extracting features of increasing complexity and abstraction. Sensitivity to shifts and distortions can be reduced by using lower-resolution feature maps in the higher layers. This is achieved by inserting subsampling layers between the convolution layers. It is important to stress that *all* the weights in such a network are trained by gradient descent. Computing the gradient can be done with a slightly modified version of the classical backpropagation procedure. The training process causes convolutional networks to automatically synthesize their own features. 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 combination of business checks and personal checks. Existing heuristic algorithms have been reimplemented as graph transformers. The modular graph transformer architecture effectively separates the global system design and the application specific algorithms.

The neural network classifier was initally trained on 500,000 images of normalized character images from various origin spanning the entire printable ASCII set. This contained both handwritten and machine-printed characters. Additional images were generated by randomly distorting the original images using simple affine transformations of the images. The network was then specialized of character images hand-segmented from check images. It was then inserted in the check reading system and trained globally.

The performance of the system at the check level is 1% error, 50% correct, and 49% rejected. 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 other commercial 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 since then.

Because it is trainable, the system can be quickly adapted to new tasks, such as reading checks from other countries or reading other types of documents.

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. A Check Reading System, based on these ideas was built. It is presently deployed commercially and reads millions of personal checks and business checks per month with record accuracy.

Although this paper presents a small number of examples of 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 scene analysis systems. 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

Bengio, Y. and Frasconi, P. (1996). An Input/Output HMM Architecture. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 427–434. MIT Press, Cambridge, MA.

Bengio, Y., LeCun, Y., Nohl, C., and Burges, C. (1995a). LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. *Neural Computation*, 7(5).

Bengio, Y., LeCun, Y., Nohl, C., and Burges, C. (1995b). LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. *Neural Computation*, 7(6).

Bottou, L. and Gallinari, P. (1991). A Framework for the Cooperation of Learning Algorithms. In Touretzky, D. and Lippmann, R., editors, *Advances in Neural Information Processing Systems*, volume 3, Denver. Morgan Kaufmann.

Bourlard, H. and Wellekens, C. (1989). Links between Markov models and multilayer perceptrons. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 1, pages 186–187, Denver. Morgan-Kaufmann.

Guyon, I., Schenkel, M., and Denker, J. (1996). Overview and synthesis of on-line cursive handwriting recognition techniques. In Wang, P. S. P. and H., B., editors, *Handbook on Optical Character Recognition and Document Image Analysis*. World Scientific.

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

Le Cun, Y. and Bengio, Y. (1995). Convolutional Networks for Images, Speech, and Time-Series. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990a). Back-Propagation Applied to Handwritten Zipcode Recognition. *Neural Computation*, 1(4).

Le Cun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and Vapnik, V. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks.*

Le Cun, Y., Matan, O., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., and Baird, H. S. (1990b). Handwritten Zip Code Recognition with Multilayer Networks. In IAPR, editor, *Proc. of the International Conference on Pattern Recognition*, Atlantic City. IEEE. invited paper.

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

Rabiner, L. R. (1989). A Tutorial On Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, volume I, pages 318–362. Bradford Books, Cambridge, MA.

Vaillant, R., Monrocq, C., and Le Cun, Y. (1994). Original approach for the localisation of objects in images. *IEE Proc on Vision, Image, and Signal Processing*, 141(4).