arXiv:1505.00393v3 [cs.CV] 23 Jul 2015

ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks

Francesco Visin[⋆]
Politecnico di Milano

Kyle Kastner[⋆]
University of Montreal

Kyunghyun Cho[⋆]
University of Montreal

Matteo Matteucci
Politecnico di Milano

Aaron Courville
University of Montreal

Yoshua Bengio
University of Montreal
CIFAR Senior Fellow

Abstract

In this paper, we propose a deep neural network architecture for object recognition based on recurrent neural networks. The proposed network, called ReNet, replaces the ubiquitous convolution+pooling layer of the deep convolutional neural network with four recurrent neural networks that sweep horizontally and vertically in both directions across the image. We evaluate the proposed ReNet on three widely-used benchmark datasets; MNIST, CIFAR-10 and SVHN. The result suggests that ReNet is a viable alternative to the deep convolutional neural network, and that further investigation is needed.

1 Introduction

Convolutional neural networks [CNN, Fukushima, 1980, LeCun et al., 1989] have become the method of choice for object recognition [see, e.g., Krizhevsky et al., 2012]. They have proved to be successful at a variety of benchmark problems including, but not limited to, handwritten digit recognition [see, e.g., Ciresan et al., 2012b], natural image classification [see, e.g., Lin et al., 2014, Simonyan and Zisserman, 2015, Szegedy et al., 2014], house number recognition [see, e.g., Goodfellow et al., 2014], traffic sign recognition [see, e.g., Ciresan et al., 2012a], as well as for speech recognition [see, e.g., Abdel-Hamid et al., 2012, Sainath et al., 2013, Tóth, 2014]. Furthermore, image representations from CNNs trained to recognize objects on a large set of more than one million images [Simonyan and Zisserman, 2015, Szegedy et al., 2014] have been found to be extremely helpful in performing other computer vision tasks such as image caption generation [see, e.g., Vinyals et al., 2014, Xu et al., 2015], video description generation [see, e.g., Yao et al., 2015] and object localization/detection [see, e.g., Sermanet et al., 2014].

While the CNN has been especially successful in computer vision, recurrent neural networks (RNN) have become the method of choice for modeling sequential data, such as text and sound. Natural language processing (NLP) applications include language modeling [see, e.g., Mikolov, 2012], and machine translation [Sutskever et al., 2014, Cho et al., 2014, Bahdanau et al., 2015]. Other popular areas of application include offline handwriting recognition/generation [Graves and Schmidhuber, 2009, Graves et al., 2008, Graves, 2013] and speech recognition [Chorowski et al., 2014, Graves and Jaitly, 2014]. RNNs have also been used together with CNNs in speech recognition [Sainath et al., 2015]. The recent revival of RNNs has largely been due to advances in learning algorithms [Pascanu et al., 2013, Martens and Sutskever, 2011] and model architectures [Pascanu et al., 2014, Hochreiter and Schmidhuber, 1997, Cho et al., 2014].

⋆ Equal contribution

The architecture proposed here is related and inspired by this earlier work, but our model relies on purely uni-dimensional RNNs coupled in a novel way, rather than on a multi-dimensional RNN. The basic idea behind the proposed ReNet architecture is to replace each convolutional layer (with convolution+pooling making up a layer) in the CNN with four RNNs that sweep over lower-layer features in different directions: (1) bottom to top, (2) top to bottom, (3) left to right and (4) right to left. The recurrent layer ensures that each feature activation in its output is an activation at the specific location *with respect to the whole image*, in contrast to the usual convolution+pooling layer which only has a local context window. The lowest layer of the model sweeps over the input image, with subsequent layers operating on extracted representations from the layer below, forming a hierarchical representation of the input.

Graves and Schmidhuber [2009] have demonstrated an RNN-based object recognition system for offline Arabic handwriting recognition. The main difference between ReNet and the model of Graves and Schmidhuber [2009] is that we use the usual sequence RNN, instead of the multidimensional RNN. We make the latter two parts of a single layer, usually (horizontal) RNNs or one (horizontal) bidirectional RNN, work on the hidden states computed by the first two (vertical) RNNs, or one (vertical) bidirectional RNN. This allows us to use a plain RNN, instead of the more complex multidimensional RNN, while making each output activation of the layer be computed with respect to the whole input image.

One important consequence of the proposed approach compared to the multidimensional RNN is that the number of RNNs at each layer scales now linearly with respect to the number of dimensions d of the input image ($2d$). A multidimensional RNN, on the other hand, requires the exponential number of RNNs at each layer (2^d). Furthermore, the proposed variant is more easily parallelizable, as each RNN is dependent only along a horizontal or vertical sequence of patches. This architectural distinction results in our model being much more amenable to distributed computing than that of Graves and Schmidhuber [2009].

In this work, we test the proposed ReNet on several widely used object recognition benchmarks, namely MNIST [LeCun et al., 1999], CIFAR-10 [Krizhevsky and Hinton, 2009] and SVHN [Netzer et al., 2011]. Our experiments reveal that the model performs comparably to convolutional neural networks on all these datasets, suggesting the potential of RNNs as a competitive alternative to CNNs for image related tasks.

2 Model Description

Let us denote by $X = \{x_{i,j}\}$ the input image or the feature map from the layer below, where $X \in \mathbb{R}^{w \times h \times c}$ with w, h and c the width, height and number of channels, or the feature dimensionality, respectively. Given a receptive field (or patch) size of $w_p \times h_p$, we split the input image X into a set of $I \times J$ (non-overlapping) patches $P = \{p_{i,j}\}$, where $I = \frac{w}{w_p}$, $J = \frac{h}{h_p}$ and $p_{i,j} \in \mathbb{R}^{w_p \times h_p \times c}$ is the (i,j)-th patch of the input image. The first index i is the horizontal index and the other index j is the vertical index.

First, we sweep the image vertically with two RNNs, with one RNN working in a bottom-up direction and the other working in a top-down direction. Each RNN takes as an input one (flattened) patch at a time and updates its hidden state, working *along each column* j of the split input image X.

$$v^F_{i,j} = f_{\text{VFWD}}(z^F_{i,j-1}, p_{i,j}), \text{ for } j = 1, \cdots, J \quad (1)$$

$$v^R_{i,j} = f_{\text{VREV}}(z^R_{i,j+1}, p_{i,j}), \text{ for } j = J, \cdots, 1 \quad (2)$$

Note that f_{VFWD} and f_{VREV} return the activation of the recurrent hidden state, and may be implemented either as a simple $\tanh$ layer, as a gated recurrent layer [Cho et al., 2014] or as a long short-term memory layer [Hochreiter and Schmidhuber, 1997].

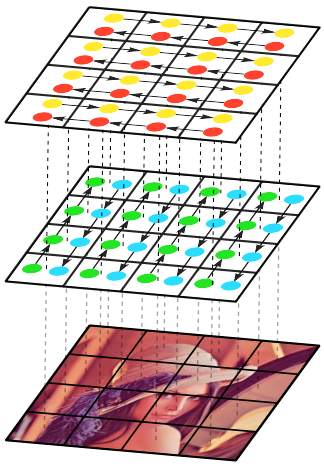

Figure 1: A one-layer ReNet

After this vertical, bidirectional sweep, we concatenate the intermediate hidden states $v^F_{i,j}$ and $v^R_{i,j}$ at each location (i, j) to get a composite feature map $V = \{v_{i,j}\}^{j=1,\ldots,J}_{i=1,\ldots,I}$, where $v_{i,j} \in \mathbb{R}^{2d}$ and d is the number of recurrent units. Each $v_{i,j}$ is now the activation of a feature detector at the location (i, j) with respect to all the patches in the j-th column of the original input ($p_{i,j}$ for all i).

Next we sweep over the obtained feature map V horizontally with two RNNs (f_{HFWD} and f_{HREV}). In a similar manner as the vertical sweep, these RNNs work along each row of V resulting in the output feature map $H = \{h_{i,j}\}$, where $h_{i,j} \in \mathbb{R}^{2d}$. Now, each vector $h_{i,j}$ represents the features of the original image patch $p_{i,j}$ *in the context of the whole image*.

Let us denote by ϕ the function from the input image map of X to the output feature map H (see Fig. 1 for a graphical illustration.) Clearly, we can stack multiple ϕ's to make the proposed ReNet deeper and capture increasingly complex features of the input image. After any number of recurrent layers are applied to an input image, the activation at the last recurrent layer may be flattened and fed into a differentiable classifier. In our experiments we used several fully-connected layers followed by a softmax classifier (as shown in Fig. 2).

The deep ReNet is a smooth, continuous function, and the parameters (those from the RNNs as well as from the fully-connected layers) can be estimated by the stochastic gradient descent algorithm with the gradient computed by backpropagation algorithm [see, e.g., Rumelhart et al., 1986] to maximize the log-likelihood.

3 Differences between LeNet and ReNet

There are many similarities and differences between the proposed ReNet and a convolutional neural network. In this section we use LeNet to refer to the canonical convolutional neural network as shown by LeCun et al. [1989]. Here we highlight a few key points of comparison between ReNet and LeNet.

At each layer, both networks apply the same set of filters to patches of the input image or of the feature map from the layer below. ReNet, however, propagates information through lateral connections that span across the whole image, while LeNet exploits local information only. The lateral connections should help extract a more compact feature representation of the input image at each layer, which can be accomplished by the lateral connections removing/resolving redundant features at different locations of the image. This should allow ReNet resolve small displacements of features across multiple consecutive patches.

LeNet max-pools the activations of each filter over a small region to achieve local translation invariance. In contrast, the proposed ReNet does not use any pooling due to the existence of learned lateral connections. The lateral connection in ReNet can emulate the local competition among features induced by the max-pooling in LeNet. This does not mean that it is not possible to use max-pooling in ReNet. The use of max-pooling in the ReNet could be helpful in reducing the dimensionality of the feature map, resulting in lower computational cost.

Max-pooling as used in LeNet may prove problematic when building a convolutional autoencoder whose decoder is an inverse[1] of LeNet, as the max operator is not invertible. The proposed ReNet is end-to-end smooth and differentiable, making it more suited to be used as a decoder in the autoencoder or any of its probabilistic variants [see, e.g., Kingma and Welling, 2014].

In some sense, each layer of the ReNet can be considered as a variant of a usual convolution+pooling layer, where pooling is replaced with lateral connections, and convolution is done without any overlap. Similarly, Springenberg et al. [2014] recently proposed a variant of a usual LeNet which does not use any pooling. They used convolution with a larger stride to compensate for the lack of dimensionality reduction by pooling at each layer. However, this approach still differs from the proposed ReNet in the sense that each feature activation at a layer is only with respect to a subset of the input image rather than the whole input image.

The main disadvantage of ReNet is that it is not easily parallelizable, due to the sequential nature of the recurrent neural network (RNN). LeNet, on the other hand, is highly parallelizable due to the independence of computing activations at each layer. The introduction of sequential, lateral

[1] All the forward arrows from the input to the output in the original LeNet are reversed.

connections, however, may result in more efficient parametrization, requiring a smaller number of parameters with overall fewer computations, although this needs to be further explored. We note that this limitation on parallelization applies only to model parallelism, and any technique for data parallelism may be used for both the proposed ReNet and the LeNet.

4 Experiments

4.1 Datasets

We evaluated the proposed ReNet on three widely-used benchmark datasets; MNIST, CIFAR-10 and the Street View Housing Numbers (SVHN). In this section we describe each dataset in detail.

MNIST The MNIST dataset [LeCun et al., 1999] consists of 70,000 handwritten digits from 0 to 9, centered on a 28×28 square canvas. Each pixel represents the grayscale in the range of $[0, 255]$.[2] We split the dataset into 50,000 training samples, 10,000 validation samples and 10,000 test samples, following the standard split.

CIFAR-10 The CIFAR-10 dataset [Krizhevsky and Hinton, 2009] is a curated subset of the 80 million tiny images dataset, originally released by Torralba et al. [2008]. CIFAR-10 contains 60,000 images each of which belongs to one of ten categories; airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each image is 32 pixels wide and 32 pixels high with 3 color channels (red, green and blue.) Following the standard procedure, we split the dataset into 40,000 training, 10,000 validation and 10,000 test samples. We applied zero-phase component analysis (ZCA) and normalized each pixel to have zero-mean and unit-variance across the training samples, as suggested by Krizhevsky and Hinton [2009].

Street View House Numbers The Street View House Numbers (SVHN) dataset [Netzer et al., 2011] consists of cropped images representing house numbers captured by Google StreetView vehicles as a part of the Google Maps mapping process. These images consist of digits 0 through 9 with values in the range of [0, 255] in each of 3 red-green-blue color channels. Each image is 32 pixels wide and 32 pixels high giving a sample dimensionality (32, 32, 3). The number of samples we used for training, valid, and test sets is 543,949, 60,439, and 26,032 respectively. We normalized each pixel to have zero-mean and unit-variance across the training samples.

4.1.1 Data Augmentation

It has been known that augmenting training data often leads to better generalization [see, e.g., Krizhevsky et al., 2012]. We decided to employ two primary data augmentations in the following experiments: *flipping* and *shifting*.

For flipping, we either flipped each sample horizontally with 25% chance, flipped it vertically with 25% chance, or left it unchanged. This allows lets the model observe "mirror images" of the original image during training. In the case of shifting, we either shifted the image by 2 pixels to the left (25% chance), 2 pixels to the right (25% chance) or left it as it was. After this first processing, we further either shifted it by 2 pixels to the top (25% chance), 2 pixels to the bottom (25% chance) or left it as it was. This two-step procedure makes the model more robust to slight shifting of an object in the image. The shifting was done without padding the borders of the image, preserving the original size but dropping the pixels which are shifted out of the input while shifting in zeros.

The choice of whether to apply these augmentation procedures on each dataset was chosen on a per-case basis in order to maximize validation performance.

4.2 Model Architectures

Gated Recurrent Units Gated recurrent units [GRU, Cho et al., 2014] and long short-term memory units [LSTM, Hochreiter and Schmidhuber, 1997] have been successful in many applications using recurrent neural networks [see, e.g., Cho et al., 2014, Sutskever et al., 2014, Xu et al., 2015].

[2] We scaled each pixel by dividing it with 255.

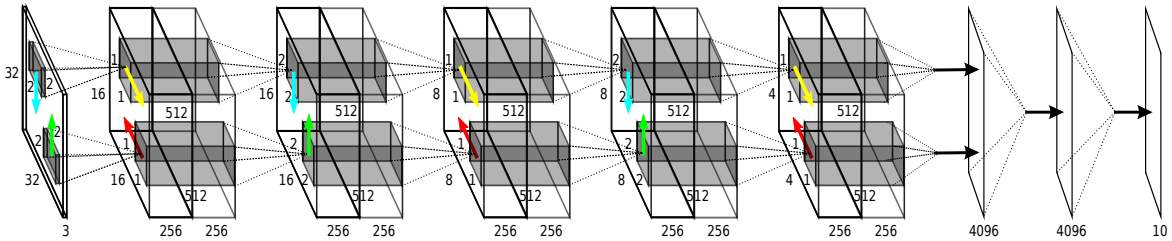

Figure 2: The ReNet network used for SVHN classification

	MNIST	CIFAR-10	SVHN
N_{RE}	2	3	3
$w_p \times h_p$	$[2 \times 2]$–$[2 \times 2]$	$[2 \times 2]$–$[2 \times 2]$–$[2 \times 2]$	$[2 \times 2]$–$[2 \times 2]$–$[2 \times 2]$
d_{RE}	256–256	320–320–320	256–256–256
N_{FC}	2	1	2
d_{FC}	4096–4096	4096	4096–4096
f_{FC}	$\max(0, x)$	$\max(0, x)$	$\max(0, x)$
Flipping	no	yes	no
Shifting	yes	yes	yes

Table 1: Model architectures used in the experiments. Each row shows respectively the number of ReNet layers, the size of the patches, the number of neurons of each ReNet layer, the number of fully connected layers, the number of neurons of the fully connected layers, their activation function and the data augmentation procedure employed.

To show that the ReNet model performs well independently of the specific implementation of the recurrent units, we decided to use the GRU on MNIST and CIFAR-10, with LSTM units on SVHN.

The hidden state of the GRU at time t is computed by

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t,$$

where

$$\tilde{h}_t = \tanh\left(W x_t + U(r_t \odot h_{t-1}) + b\right)$$

and

$$[u_t; r_t] = \sigma\left(W_g x_t + U_g h_{t-1} + b_g\right).$$

For more details on the LSTM unit, as well as for an in-depth comparison among different recurrent units, we refer the reader to [Chung et al., 2015].

General Architecture The principal parameters that define the architecture of the proposed ReNet are the number of ReNet layers (N_{RE}), their corresponding receptive field sizes ($w_p \times h_p$) and feature dimensionality (d_{RE}), the number of fully-connected layers (N_{FC}) and their corresponding numbers (d_{FC}) and types (f_{FC}) of hidden units.

In this introductory work, we did not focus on extensive hyperparameter search to find the optimal validation set performance. We chose instead to focus the experiments on a small set of hyperparameters, with the only aim to show the potential of the proposed model. Refer to Table 1 for a summary of the settings that performed best on the validation set of the studied datasets and to Fig. 2 for a graphical illustration of the model we selected for SVHN.

4.3 Training

To train the networks we used a recently proposed adaptive learning rate algorithm, called Adam [Kingma and Ba, 2014]. In order to reduce overfitting we applied dropout [Srivastava et al., 2014] after each layer, including both the proposed ReNet layer (after the horizontal and vertical

sweeps) and the fully-connected layers. The input was also corrupted by masking out each variable with probability 0.2. Finally, each optimization run was early stopped based on validation error.

Note that unlike many previous works, we did not retrain the model (selected based on the validation performance) using both the training and validation samples. This experiment design choice is consistent with our declared goal to show a proof of concept rather than stressing absolute performance. There are many potential areas of exploration for future work.

Test Error	Model
0.28%	[Wan et al., 2013]⋆
0.31%	[Graham, 2014a]⋆
0.35%	[Ciresan et al., 2010]
0.39%	[Mairal et al., 2014]⋆
0.39%	[Lee et al., 2014]⋆
0.4%	[Simard et al., 2003]⋆
0.44%	[Graham, 2014b]⋆
0.45%	[Goodfellow et al., 2013]⋆
0.45%	**ReNet**
0.47%	[Lin et al., 2014]⋆
0.52%	[Azzopardi and Petkov, 2013]

(a) MNIST

Test Error	Model
4.5%	[Graham, 2014b]⋆
6.28%	[Graham, 2014a]⋆
8.8%	[Lin et al., 2014]⋆
9.35%	[Goodfellow et al., 2013]⋆
9.39%	[Springenberg and Riedmiller, 2013]⋆
9.5%	[Snoek et al., 2012]⋆
11%	[Krizhevsky et al., 2012]⋆
11.10%	[Wan et al., 2013]⋆
12.35%	**ReNet**
15.13%	[Zeiler and Fergus, 2013]⋆
15.6%	[Hinton et al., 2012]⋆

(b) CIFAR-10

Test Error	Model
1.92%	[Lee et al., 2014]⋆
2.23%	[Wan et al., 2013]⋆
2.35%	[Lin et al., 2014]⋆
2.38%	**ReNet**
2.47%	[Goodfellow et al., 2013]⋆
2.8%	[Zeiler and Fergus, 2013]⋆

(c) SVHN

Table 2: Generalization errors obtained by the proposed ReNet along with those reported by previous works on each of the three datasets. ⋆ denotes a convolutional neural network. We only list the results reported by a single model, i.e., no ensembling of multiple models. In the case of SVHN, we report results from models trained on the Format 2 (cropped digit) dataset only.

5 Results and Analysis

In Table 2, we present the results on three datasets, along with previously reported results.

It is clear that the proposed ReNet performs comparably to deep convolutional neural networks which are the *de facto* standard for object recognition. This suggests that the proposed ReNet is a viable alternative to convolutional neural networks (CNN), even on tasks where CNNs have historically dominated. However, it is important to notice that the proposed ReNet does not outperform state-of-the-art convolutional neural networks on any of the three benchmark datasets, which calls for more research in the future.

6 Discussion

Choice of Recurrent Units Note that the proposed architecture is independent of the chosen recurrent units. We observed in preliminary experiments that gated recurrent units, either the GRU or the LSTM, outperform a usual sigmoidal unit (affine transformation followed by an element-wise sigmoid function.) This indirectly confirms that the model utilizes long-term dependencies across an input image, and the gated recurrent units help capture these dependencies.

Analysis of the Trained ReNet In this paper, we evaluated the proposed ReNet only quantitatively. However, the accuracies on the test sets do not reveal what kind of image structures the ReNet has captured in order to perform object recognition. Due to the large differences between ReNet and LeNet discussed in Sec. 3, we expect that the internal behavior of ReNet will differ from that of LeNet significantly. Further investigation along the line of [Zeiler and Fergus, 2014] will be needed, as well exploring ensembles which combine RNNs and CNNs for bagged prediction.

Computationally Efficient Implementation As discussed in Sec. 3, the proposed ReNet is less parallelizable due to the sequential nature of the recurrent neural network (RNN). Although this sequential nature cannot be addressed directly, our construction of ReNet allows the forward and backward RNNs to be run independently from each other, which allows for parallel computation. Furthermore, we can use many parallelization tricks widely used for training convolutional neural networks such as parallelizing fully-connected layers [Krizhevsky, 2014], having separate sets of kernels/features in different processors [Krizhevsky et al., 2012] and exploiting data parallelism.

Acknowledgments

The authors would like to thank the developers of Theano [Bergstra et al., 2010, Bastien et al., 2012]. We acknowledge the support of the following organizations for research funding and computing support: NSERC, Samsung, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. F.V. was funded by the AI*IA Young Researchers Mobility Grant and the Politecnico di Milano PHD School International Mobility Grant.

References

O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012.

G. Azzopardi and N. Petkov. Trainable COSFIRE filters for keypoint detection and pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2):490–503, 2013.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR 2015)*, 2015.

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. Submited to the Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, Oct. 2014. to appear.

J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. arXiv:1412.1602, 2014.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, 2015.

D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012a.

D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. Technical report, arXiv:1202.2745, 2012b.

D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep big simple neural nets excel on handwritten digit recognition. *arXiv*, abs/1003.0358, 2010.

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327, 2013.

I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In *International Conference on Learning Representations*, 2014.

B. Graham. Spatially-sparse convolutional neural networks. *arXiv*, abs/1409.6070, 2014a.

B. Graham. Fractional max-pooling. *arXiv*, abs/1412.6071, 2014b.

A. Graves. Generating sequences with recurrent neural networks. Technical report, arXiv preprint arXiv:1308.0850, 2013.

A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML'2014*, 2014.

A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS'2008*, pages 545–552. 2009.

A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS'2007*, pages 577–584. 2008.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*, abs/1207.0580, 2012.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:*`1412.6980 [cs.LG]`, Dec. 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS'2012)*. 2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–345. Springer, 1999.

C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv*, abs/1409.5185, 2014.

M. Lin, Q. Chen, and S. Yan. Network in network. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, Apr. 2014.

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2627–2635, 2014.

J. Martens and I. Sutskever. Learning recurrent neural networks with Hessian-free optimization. In *Proc. ICML'2011*. ACM, 2011.

T. Mikolov. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. Deep Learning and Unsupervised Feature Learning Workshop, NIPS, 2011.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML'2013*, 2013.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. In *ICLR'2014*, 2014.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986.

T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013.

T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, 2015.

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *International Conference on Learning Representations*, 2014.

P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*, pages 958–962, 2003.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR 2015)*, 2015.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2960–2968, 2012.

J. T. Springenberg and M. A. Riedmiller. Improving deep neural networks with probabilistic maxout units. *arXiv*, abs/1312.6116, 2013.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS'2014*, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

L. Tóth. Combining time-and frequency-domain convolution in convolutional neural network-based phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 190–194. IEEE, 2014.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: a neural image caption generator. arXiv 1411.4555, 2014.

L. Wan, M. D. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1058–1066, 2013.

K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. arXiv:1502.03044, 2015.

L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Video description generation incorporating spatio-temporal features and a soft-attention mechanism. *arXiv:1502.08029*, 2015.

M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv*, abs/1301.3557, 2013.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV'14*, 2014.