Deep Convolutional Networks with Non-convolutional Recurrent Memory for Structured Prediction

Joel Ruben Antony Moniz [1 2 3] **Christopher Pal** [1 2]

Abstract

Very deep convolutional neural networks (CNNs) yield state of the art results on a wide variety of visual recognition problems. Recent evidence also indicates that convolutional networks could benefit from an interface to recurrent neural networks (RNNs) with explicitly constructed memory mechanisms, which would help address the challenges of structured prediction by allowing for a form of iterative inference. We propose and evaluate a memory mechanism enhanced convolutional neural network architecture based on augmenting residual networks (ResNets) with a non-convolutional long short term memory mechanism. Our experiments yield results comparable to state-of-the-art results on a structured prediction task (specifically, face keypoint detection), but with much fewer parameters and a significantly lower computational cost. Interestingly, we achieve this with an extremely general-purpose architecture which also yields results among the top performing methods in an entirely different domain- classification- on the CIFAR-100 and CIFAR-10 datasets.

1. Introduction

Deep convolutional neural networks have recently exceeded human level performance on a number of tasks. As an example, residual networks (ResNets) have taken the top spot on the standard ImageNet evaluation, exceeding human performance in terms of the key top-5 error rate (He et al., 2015). Well known designs for deep convolutional neural network architectures such as ResNets and VGG architectures (Simonyan & Zisserman, 2014) currently do not have explicitly designed recurrent computation or memory mechanisms. However, as deep learning techniques are being applied to more and more complex visual recognition problems, the ability of CNNs to interface with recurrent processing elements and to manipulate memory contents outside of the traditional CNN processing pipeline becomes a seductive property. Our contribution is to propose and explore a novel CNN architecture with an interface to non-convolutional recurrent processing and memory manipulation elements. Here, instead of using a deep LSTM network as a mechanism that processes elements in a time series or in a specific pattern or sequence, we use its inputs as the abstract feature representations of a very deep CNN. In this way the LSTM mediates an interface between the recurrent processing elements, internal memory and the algorithmic state of an LSTM RNN with the features created by the layers of a CNN.

When used for structured predictions, each input step consists of features representing higher levels of abstraction in the CNN feature hierarchy, while each step of the RNN may serve as an iterative inference regarding the underlying structured prediction problem. This formulation thus not only affords such models the capacity to perform memory manipulation, but also provides alternative general purpose algorithmic processing operations that execute along side classical convolutional network processing units.

As shown in the results section, we find that this architecture affords a significant reduction in the number of parameters and the computational cost over the state-of-the-art method (Recombinator Networks (Honari et al., 2016)), while increasing the loss on the 300W dataset (Sagonas et al., 2013) only marginally. We also find that the CRMN performs better than various ResNet baselines, even those with a significantly larger number of parameters and floating point operations. Remarkably, and for the first time to the best of the authors' knowledge, this architecture is extremely general, and also provides results comparable to the best performing results at the time of writing in an entirely different domain – image classification – on the CIFAR-10 and CIFAR-100 datasets (Krizhevsky & Hinton, 2009). An earlier version of this work may be found at (Moniz & Pal, 2016).

[1]The Montreal Institute for Learning Algorithms [2]Polytechnique Montreal [3]Birla Institute of Technology and Science, Pilani. Correspondence to: Christopher Pal <christopher.pal@polymtl.ca>.

Proceedings of the ICML 17 Workshop on Deep Structured Prediction, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

2. Related Work

As deep networks optimized with gradient descent must deal with the issue of vanishing or exploding gradients, various architectures and mechanisms have been proposed to address the underlying issue. Deep residual networks have recently emerged as an extremely popular architecture that mitigates the valishing/exploding gradient issue even for extremely deep netowrks, and are among the top performing architectures on the ImageNet, CIFAR-10 and CIFAR-100 benchmarks. Here, the residual connections composed of skip connections that are combined additively with the output of convolutional blocks of are used.

Recurrent neural networks (RNNs) also suffer from similar issues with vanishing and exploding gradients. LSTM RNNs were explicitly designed to deal with the issue through the manipulation of a memory cell having the property that gradient information can be captured and stored without degradation over many timesteps – or in our case, many network layers.

While recurrent convolutional networks (RCNs) have been used for video processing, typically feeding information from 2D CNNs into an RNN which creates the representation for the video, a recently proposed method for video processing (Ballas et al., 2015) has integrated CNNs with gated recurrent units or GRUs (Chung et al., 2014) both temporally and along the feature map hierarchy to form a stacked GRU-RCN. While this network is potentially very deep in terms of the temporal dimension, at 5 layers, it is shallow in terms of per frame abstraction depth. The positive impact of having a GRU recurrent network along the abstraction depth of a VGG grade CNN points to potential advantages of using more sophisticated RNN mechanisms along the feature abstraction hierarchy of extremely deep CNN architectures.

The Highway Network (Srivastava et al., 2015) architecture consists of a mechanism allowing 2D-CNNs to interact with a simple memory mechanism. In the case of a convolutional highway network, input layers are transformed using convolutional layers, the elements of which are either passed or blocked using a gating mechanism controlled by *Transform* and *Carry* Gates, which themselves are learned functions of the input. Highway Networks were found to not suffer from increasing depth, and converge much better than a regular deep CNN. This work further substantiates the notion that explicit memory mechanisms interacting with CNNs could be a potent combination.

Recombinator Networks (Honari et al., 2016) proposes a 2D convolutional structure that aims to leverage both coarse and fine grained features to accurately perform the structured prediction of facial keypoints, along with a denoising network to ensure that the system's predicted keypoints conform to the typical keypoint layout. The architecture proposed points strongly towards the possible benefit of using a recurrent mechanism to help the network remember important coarse grained features (in the form of the features in the initial layers of the network), and combining these coarse-grained features with the fine-grained features typically observed at the output end of a deep network.

3. Convolutional Residual Memory Networks

The CRMN architecture (Figure 1) we propose drops an LSTM on top of a standard convnet architecture (in this case, a ResNet). Thus, the LSTM in our CRMN architecture takes views of the abstraction hierarchy as input, instead of a time series, with features increasing in abstraction as the series progresses (as opposed to changing in time). Before passing each feature map into the LSTM, we apply a meanpool sub-sampling on the feature map (which both cuts down the total number of parameters and training time, and also improves performance, likely due to the easier optimization of the now much lighter LSTM) and then flatten it. When the size of a flattened feature map is less than the maximum size (after the first ResNet block, when a convolution of stride 2 has been used), the features are zero-padded. Both the final hidden state output of the LSTM and the output of the global average pooling layer at the end of the ResNet are fed into a fully connected layer.

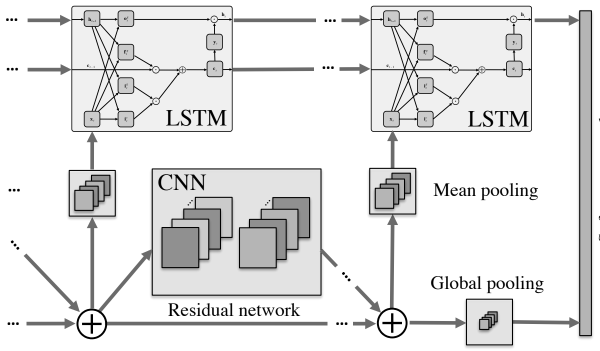

Figure 1. The repeating computational block and the final computational units of a convolutional residual memory network (CRMN).

More precisely, we begin with a ResNet computational block following the formulation in (He et al., 2015). Define $\boldsymbol{x}_l$ as the set of all convolutional feature maps for layer l and $\boldsymbol{W}_l$ as all parameters for the layer. We use the output of a ResNet with 2-layer convolutional blocks denoted by the residual mapping function $\boldsymbol{f}(\boldsymbol{x}_l, \{\boldsymbol{W}_l\})$) along with the identity mapping as the input to each LSTM unit after the element-wise addition with the previous layer. This implies the following formulation for our CRMN, where

$\boldsymbol{x}_l = \boldsymbol{x}_{l-1} + \boldsymbol{f}(\boldsymbol{x}_{l-1}, \{\boldsymbol{W}_l\})$, is the input at each LSTM interface, step l. As discussed above, we reduce the size of this input using mean pooling. Using $\boldsymbol{W}_x$ and $\boldsymbol{b}_x$ to denote the various weight matrices and bias terms, the input, forget and output gates for the LSTM are then given by

$$\begin{aligned}\boldsymbol{i}_l &= \boldsymbol{\sigma}_{iof}(\boldsymbol{W}_{xi}\boldsymbol{x}_l + \boldsymbol{W}_{hi}\boldsymbol{h}_{l-1} + \boldsymbol{w}_{ci} \odot \boldsymbol{c}_{t-1} + \boldsymbol{b}_i),\\ \boldsymbol{f}_l &= \boldsymbol{\sigma}_{iof}(\boldsymbol{W}_{xf}\boldsymbol{x}_l + \boldsymbol{W}_{hf}\boldsymbol{h}_{l-1} + \boldsymbol{w}_{cf} \odot \boldsymbol{c}_{l-1} + \boldsymbol{b}_f),\\ \boldsymbol{o}_l &= \boldsymbol{\sigma}_{iof}(\boldsymbol{W}_{xo}\boldsymbol{x}_l + \boldsymbol{W}_{ho}\boldsymbol{h}_{l-1} + \boldsymbol{w}_{co} \odot \boldsymbol{c}_l + \boldsymbol{b}_o),\end{aligned}$$

where we use the $logistic$ function as the nonlinearity for the input, forget and output gates $\boldsymbol{\sigma_{iof}}$. The peephole connections are given by the terms of the form $\boldsymbol{w}_x \odot \boldsymbol{c}_t$. The cell gates are given by

$$\begin{aligned}\boldsymbol{c}_l &= \boldsymbol{f}_l \odot \boldsymbol{c}_{l-1} + \boldsymbol{i}_l \odot \boldsymbol{s}_l,\\ \boldsymbol{s}_l &= \boldsymbol{\sigma}_c(\boldsymbol{W}_{xc}\boldsymbol{x}_l + \boldsymbol{W}_{hc}\boldsymbol{h}_{l-1} + \boldsymbol{b}_c),\end{aligned}$$

where we use $tanh$ as the cell gate non-linearity $\boldsymbol{\sigma_c}$. The final output for the hidden unit update is given by

$$\boldsymbol{h}_l = \boldsymbol{o}_l \odot \boldsymbol{\sigma}_h(\boldsymbol{c}_l).$$

As discussed above and illustrated in figure 1, the final prediction is made using global pooling of the final ResNet layers and the output of the LSTM hidden state as input to a sigmoid layer (if the model is used for keypoint prediction) or a softmax prediction layer (if the model is used for classification, shown in 1).

4. Experiments

In most of our experiments, we observed that architectures trained for longer before a learning rate shift perform better on the validation set in general, so we train for at least m epochs, before the first learning rate shift, m being a hyperparameter. We make other learning rate changes a few epochs after the loss on the validation set stops decreasing. Since the architecture proposed in this paper can be divided into 3 sub-components: the Residual Network, the LSTM, and the fully connected layer, we explored a round robin learning rate (RRLR) schedule when updating the learning rate of each component sequentially, starting with the ResNet, then the LSTM, then the sigmoid layer.

4.1. Helen, LFPW and iBug Experiments

The additional LSTM mechanism in our formulation allows for a qualitatively different form of algorithmic manipulation which we conjecture can be particularly useful if the underlying task is more complicated than simple classification. When a structured prediction is desired, some form of internal inference procedure that accounts for the fact that a joint prediction must be made by the model is a desirable property and this fact in part motivated our design explored here. A number of recent papers have highlighted the relationships between recurrent networks and classical methods for structured prediction such as random fields and conditional random fields (Zheng et al., 2015). We therefore test the ability of an adapted CRMN to make a structured prediction for the well known task of localizing facial keypoints. In contrast to RCNs, here we use an LSTM which executes in parallel with the underlying CNN and which makes *continuous valued* keypoint predictions, i.e. performs multivariate non-linear regression.

We trained the model on the train sets of the AFW, Helen and LFPW datasets, and evaluated the performance on the test set (using a validation set to determine when to change the learning rate). All our hyperparameters are *exactly* as in (Honari et al., 2016), except that we use the round-robin learning schedule described above instead of the Adam optimizer (Kingma & Ba, 2014) (though we use the same starting learning rate)- we decrease the learning rate by a factor of 0.1, 0.5 and 0.2 successively.

We show how our CRMN fares vis-a-vis other methods and baselines in 1. To the best of our knowledge, the best performing method on these datasets is the Recombinator Network architecture. It is important to note that almost all other method of structured prediction and classification are fine-tuned *specifically* for the task, and would be difficult to extend to other use-cases. Further, the underlying CNN component of the models created here have not been designed for the keypoint prediction task – which is known to be important.

In brief, we find that this architecture reduces the number of parameters (by around 28%), and drastically reduces the computational cost (by a *factor* of around 23x) of the state-of-the-art method (Recombinator Networks (Honari et al., 2016)), while increasing the loss on the 300W dataset (Sagonas et al., 2013) only marginally (an increase of less than 8% on the Helen+LFPW dataset, and less than 6% on the iBug dataset). We also find that the CRMN performs slightly better than even a baseline ResNet that has over *three times* the number of parameters and takes over *seven times* more floating point operations.

Model	n	fm	Params	FLOPs	Common	iBug
RCN	-	-	24.8M	327.84G	4.67	8.44
CRMN	5	4	17.76M	14.07G	5.04	8.92
ResNet	18	4	27.63M	50.59G	5.49	9.51
ResNet	9	8	54.55M	100.30G	5.14	8.94
ResNet	9	4	13.67M	25.09G	5.40	9.45
ResNet	5	4	7.46M	13.75G	5.91	10.23

Table 1. Mean test set error normalized by interocular distance

4.2. CIFAR-100 and CIFAR-10 Experiments

The architecture descried above has not been designed specifically for the keypoint prediction task – which is known to be important – and yields extremely competitive results in the classification scenario as well, with minimal changes: we simply replaced the final sigmoid layer with a softmax layer, changed the pooling from a 4x4 pooling to a 2x2 pooling because of the smaller image size, and changed the learning rate schedule to one with learning rates [0.1, 0.01, 0.005, 0.001] (which is closer to the one proposed by (He et al., 2015)).

We outline our results on the CIFAR-100 dataset, along with the corresponding configurations we used (in terms of depth and width), in Table 2. Our CIFAR-10 results are presented in Table 3. In each table, we provide the other top performing methods on the respective datasets for context.

Accuracy (%) and Method	
67.61	Highway Network (100 layers) (Srivastava et al., 2015)
75.42	Stochastic Depth (Huang et al., 2016b)
77.28	Swapout (32 layers; 64 fm) (Singh et al., 2016)
77.29	Pre-act. ResNet (He et al., 2016) (1001 layers)
79.96	WResNet + dropout (28 layers; 160 fm) (Zagoruyko & Komodakis, 2016)
80.75	DenseNets (100 layers) (Huang et al., 2016a)
78.27	Our CRMN (32 layers; 64 fm)
79.68	Our CRMN (28 layers; 160 fm), RRLR
80.21	Our CRMN (32 layers; 192 fm), RRLR

Table 2. CIFAR-100 Accuracies.

Accuracy (%) and Method	
90.62	Maxout Network (Goodfellow et al., 2013)
92.40	Highway Network (Srivastava et al., 2015)
93.57	ResNet (110 layers) (He et al., 2015)
94.77	Stochastic Depth (Huang et al., 2016b)
95.24	Swapout (32 layers; 64 fm) (Singh et al., 2016)
95.38	Pre-act. ResNet (He et al., 2016) (1001 layers)
95.50	Frac. Max-pool (Graham, 2014) (1 test)
95.59	All Conv. Net (Springenberg et al., 2014)
95.83	WResNet (28 layers; 160 fm) (Zagoruyko & Komodakis, 2016)
96.53	Frac. Max-pool[a] (Graham, 2014) (100 tests)
96.26	DenseNets (100 layers) (Huang et al., 2016a)
95.60	Our CRMN (28 layers; 160 fm), RRLR
95.84	Our CRMN (32 layers; 192 fm[b]), RRLR

Table 3. CIFAR-10 Accuracies.

[a] An accuracy of 96.53% was obtained using the Fractional Max-pooling approach in (Graham, 2014); however, *it was obtained using 100 tests*; using a single test the method in (Graham, 2014) yielded 95.5%

[b] *fm* represents the number of feature maps in the first convolutional layer of the ResNet

5. Discussion and Conclusions

We have explored a novel deep convolutional network architecture that uses an LSTM which iterates on the increasing abstraction offered by the CNN, as opposed to a time series or a sequence as it traditionally does. Our formulation thus allows CNNs to be extended with a parallel network taking intermediate representations as input and subjecting them to alternative algorithmic manipulations of the type suitable for iterative inference and structured prediction. In a structured prediction (more specifically, keypoint detection) context, we observe that our architecture yields results comparable with the current state-of-the-art method (with only a 6-8% increase in the error), but substantially decreases the number of parameters (by around 28%). Remarkably, it radically reduces the computational cost (measured in terms of the number of floating point operations) by a *factor* of around 23x. This is especially significant with the advent of the adoption of deep learning techniques on mobile platforms, where models must particularly be computationally efficient because processing power is at a premium.

In addition, the proposed architecture is not tailor made to the keypoint prediction problem, and can easily be extended to the classification domain with almost no changes to the model. We have provided experiments on the standard CIFAR-10 and CIFAR-100 classification benchmarks using our proposed approach. We observed that our model yields results that are extremely competitive with the state-of-the-art in a classification context.

Other recent work has also underscored the connections between ResNets and RNNs, highlighting the potential computational advantages of using RNNs as well as the biological plausibility of such constructions (Zheng et al., 2015). Their work emphasized the potential to reduce the number of parameters of a ResNet using analogous RNNs. One of the main observations and conclusions from our work is that the additional RNN pipeline in our formulation allows the overall computation of a model to be reduced, and affords dramatically improved performance over similarly configured ResNet-only architectures.

ACKNOWLEDGMENTS

We are grateful to Christopher Beckham for helpful discussions and feedback. We thank the developers of Theano (Team et al., 2016) and Lasagne (Dieleman et al., 2015). We would also like to thank Samsung for supporting this research.

References

Ballas, Nicolas, Yao, Li, Pal, Chris, and Courville, Aaron. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432 and Proc. ICLR 2016*, 2015.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Dieleman, Sander, Schlter, Jan, Raffel, Colin, Olson, Eben, Snderby, Sren Kaae, Nouri, Daniel, et al. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878.

Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

Graham, Benjamin. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

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

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *arXiv preprint:1603.05027*, 2016.

Honari, Sina, Yosinski, Jason, Vincent, Pascal, and Pal, Christopher. Recombinator networks: Learning coarse-to-fine feature aggregation. In *In Proc. CVPR*, 2016.

Huang, Gao, Liu, Zhuang, and Weinberger, Kilian Q. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016a.

Huang, Gao, Sun, Yu, Liu, Zhuang, Sedra, Daniel, and Weinberger, Kilian. Deep networks with stochastic depth. *arXiv preprint arXiv:1603.09382*, 2016b.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Moniz, Joel and Pal, Christopher. Convolutional residual memory networks. *arXiv preprint arXiv:1606.05262*, 2016.

Sagonas, Christos, Tzimiropoulos, Georgios, Zafeiriou, Stefanos, and Pantic, Maja. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proc. ICCV Workshops*, pp. 397–403, 2013.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Singh, Saurabh, Hoiem, Derek, and Forsyth, David A. Swapout: Learning an ensemble of deep architectures. *arXiv preprint arXiv:1605.06465*, 2016.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806 and ICLR 2015*, 2014.

Srivastava, Rupesh K, Greff, Klaus, and Schmidhuber, Jürgen. Training very deep networks. In *Advances in Neural Information Processing Systems*, pp. 2368–2376, 2015.

Team, The Theano Development, Al-Rfou, Rami, Alain, Guillaume, Almahairi, Amjad, Angermueller, Christof, Bahdanau, Dzmitry, Ballas, Nicolas, Bastien, Frédéric, Bayer, Justin, Belikov, Anatoly, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zheng, Shuai, Jayasumana, Sadeep, Romera-Paredes, Bernardino, Vineet, Vibhav, Su, Zhizhong, Du, Dalong, Huang, Chang, and Torr, Philip HS. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.