Automate Deep Learning in Image Processing

Xinyu Ma
Institute of Computing Technology
Chinese Academy of Science
Beijing, P.R.C
xinyuma2016@gmail.com

Abstract

Convolutional neural work(CNN) have achieved great success in image processing. This kind of neural networks can recognize visual patterns directly from pixel images with minimal preprocessing. But it is hard to design the architecture and also restricted by the availability of the computing resources. For example, GoogleNet has 22 hidden layers and trained on 2 GPUS for a week; VGGNet consists of 16 convolutional layers and trained on 4 GPUs for 2-3 weeks. I did a research about automated exploring and designing a better architecture of neural networks and combining some work I have done this semester about automated exploring CNN architecture with reinforcement learning. Distributed neural network training platform is also necessary for accelerating model training.

1 Introduction

With the development of computing resource, like GPU and AI chip, machine learning experts can design deeper and more complicated neural networks. The common problem in deep learning area is that it's still very difficult to design neural network architecture and model's hyper parameter tuning is too heady and too tedious. So I am very interested in how to automatically design a good architecture of neural networks especially of convolutional neural networks in a dataset without hyper parameter tuning and other works. It means that you give a dataset, platform will give you a good neural network model.

Firstly, I will introduce some classic CNN architecture like GoogleNet, ResNet. Or maybe I should introduce CNN first. A typical CNN architecture consists of several convolution, pooling, and fully connected layers. While constructing a CNN, a network designer has to make numerous design choices: the number of layers of each type, the ordering of layers, and the hyper parameters for each type of layer, e.g., the receptive field size, stride, and number of receptive fields for a convolution layer. The number of possible choices makes the design space of CNN architectures extremely large and hence, infeasible for an exhaustive manual search. While there has been some work (Pinto et al., 2009; Bergstra et al., 2013; Domhan et al., 2015;Bowen et al., 2017;Hieu et al. 2018) on automated or computer-aided neural network design, new CNN architectures or network design elements are still primarily developed by researchers using new theoretical insights or intuition gained from experimentation.

1.1 GoogleNet

GoogleNet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2014) competition. It achieved a top-5 error rate of 6.67%. This was very close to human level performance. The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. It used batch normalization, image distortions and

RMSprop. This module is based on several very small convolutions in order to drastically reduce the number of parameters. Its architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million.

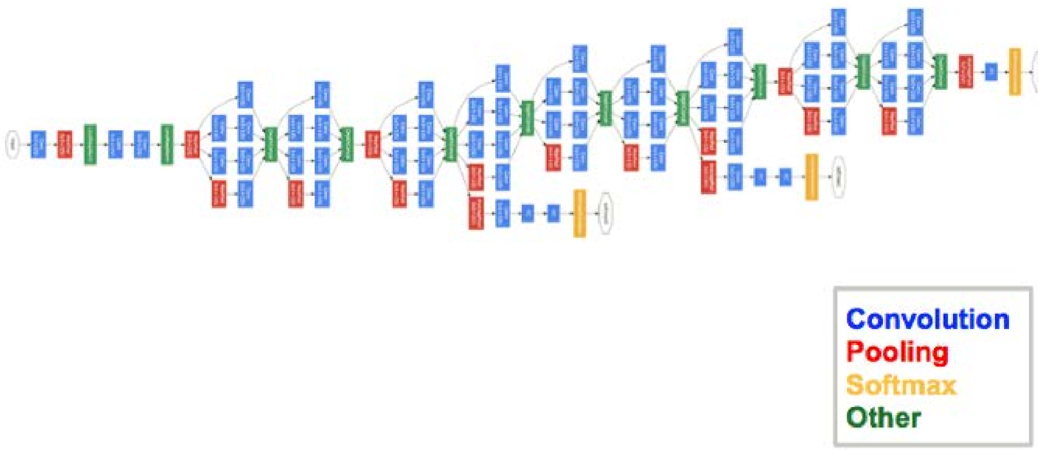

Figure 1. GoogleNet

We can see from figure 1 that GoogleNet's architecture is not only the simple combination of convolution, pooling and fully connected layers' order, it has small modules called Inception which is more complicated than all CNNs' architecture that we have ever seen.

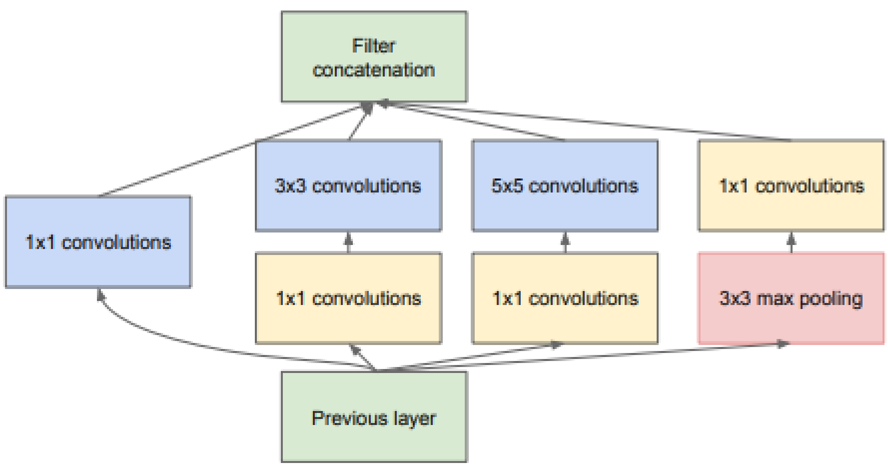

Figure 2.Inception

Inception module includes 1x1, 3x3 and 5x5 convolutions, 3x3 max pooling. They are combined in a fixed order that embodies senior deep learning expert's wisdom.

1.2 ResNet

Residual networks(ResNet) won the ILSVRC 2015 classification task by achieving 3.57% error on the ImageNet test set with an ensemble of these residual networks. This was almost above human level performance. ResNet are easier to optimize and can gain accuracy from considerably increased depth. It has 152 layers and includes 3x3 and 5x5 convolutions with different filter size.

It's impossible to train such deeper neural network with current computing resource. So ResNet proposed let layers fit a residual mapping not the original to avoid gradient exploding and vanishing. Also ResNet proposed shortcut to skip some layers and connect layers. The identity shortcuts can be directly used when the input and output are of the same dimensions. When the dimensions increase, ResNet consider two options: (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter; (B) The projection shortcut is used to match dimensions (done by 1x1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

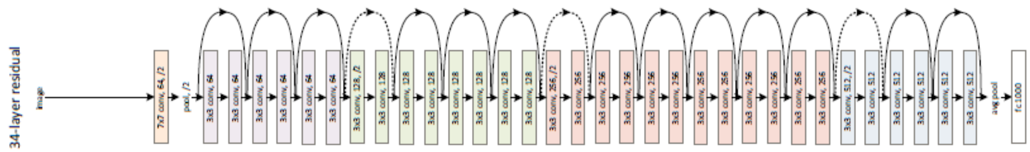

Figure 3. ResNet

As we can see, it's too complicate to design for general deep learning experts. So, here comes the question: Can we design CNN automatly?

2 Related Work

Sometimes designing neural network architectures and reinforcement learning can do independently. Research on automating neural network design goes back to the 1980s when genetic algorithm-based approaches were proposed to find both architectures and weights (Schaffer et al., 1992). However, to the best of our knowledge, networks designed with genetic algorithms, such as those generated with the NEAT algorithm (Stanley & Miikkulainen, 2002), have been unable to match the performance of hand-crafted networks on standard benchmarks (Verbancsics & Harguess, 2013). Other biologically inspired ideas have also been explored; motivated by screening methods in genetics, Pinto et al. (2009) proposed a high-throughput network selection approach where they randomly sample thousands of architectures and choose promising ones for further training. In recent work, Saxena & Verbeek (2016) propose to sidestep the architecture selection process through densely connected networks of layers, which come closer to the performance of hand-crafted networks. Google also proposed Efficient Neural Architecture Search(ENAS), a fast and inexpensive approach for automatic model design. In ENAS, a controller discovers neural network architectures by searching for an optimal subgraph within a large computational graph. The controller is trained with policy gradient to select a subgraph that maximizes the expected reward on a validation set. Details we will talk about later.

Recently there has been much work at the intersection of reinforcement learning and deep learning. For instance, methods using CNNs to approximate the Q-learning utility function (Watkins, 1989) have been successful in game-playing agents (Mnih et al., 2015; Silver et al., 2016) and robotic control (Lillicrap et al., 2015; Levine et al., 2016). These methods rely on phases of exploration, where the agent tries to learn about its environment through sampling, and exploitation, where the agent uses what it learned about the environment to find better paths. In traditional reinforcement learning settings, over-exploration can lead to slow convergence times, yet over-exploitation can lead to convergence to local minima (Kaelbling et al., 1996). However, in the case of large or continuous state spaces, the -greedy strategy of learning has been empirically shown to converge (Vermorel & Mohri, 2005). Finally, when the state space is large or exploration is costly, the experience replay technique (Lin, 1993) has proved useful in experimental settings (Adam et al., 2012; Mnih et al., 2015). Bowen incorporate these techniques—Q-learning, the ϵ-greedy strategy and experience replay—in algorithm design.

3 AutoML

Automated Machine Learning(AutoML) is developed by Google and it mainly provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning. It's amazing if people do not need to select an appropriate model family and optimize model hyper parameters. AutoML can also help you do these works: preprocess clean the data, select and construct appropriate model family, post process machine learning models and critically analyze the results obtained. Most machine learning experts will be substituted by AutoML or other automated tools for machine learning. So I decide to do some research for not be eliminated in the future. Hahaha…

AutoML did a lot of work in neural architecture search. Their team published many papers, see the following website : https://www.ml4aad.org/automl/literature-on-neural-architecture-search/.

They present an approach to automate the process of discovering optimization methods, with a focus on deep learning architectures. They train a Recurrent Neural Network controller to generate a string in a domain specific language that describes a mathematical update equation based on a list of primitive functions, such as the gradient, running average of the gradient, etc. The controller is trained with Reinforcement Learning to maximize the performance of a model after a few epochs.

3.1 Model Description

CNN consists of many layers which can be described by string sequences, like “convolution, RELU, max pooling, batch normalization …”. This is very useful for large-scale model generating by neural network. GoogleNet can be described as follows:


```
"op":"CONV:7X7:2X2:SAME",
"op":"MAX_POOLING:3X3:2X2:SAME",
"op":"LocalRespNorm",
"op":"CONV:3X3:1X1:SAME",
"op":"LocalRespNorm",
"op":"MAX_POOLING:3X3:2X2:SAME",
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":"MAX_POOLING:3X3:2X2:SAME",
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
 ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X
 3:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":"MAX_POOLING:3X3:2X2:SAME",
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":[["CONV:1X1:1X1:SAME",["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SA
ME"],["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],["MAX_POOLING:3X3
:1X1:SAME","CONV:1X1:1X1:SAME"]],"DepthContact"],
"op":"AVERAGE_POOLING:7X7:1X1:VALID",
"op":"FC: RELU",
"op":"SOFTMAX",
"regularization":"Dropout",
"coefficient":"0.4"
```


Figure 4.CNN to String Sequence

Figure 4. is an example about translating GoogleNet architecture to string sequences. GoogleNet has another name called as Inception(Figure 2.) which one inception module can be described with:

```
[["CONV:1X1:1X1:SAME",
  ["CONV:1X1:1X1:SAME","CONV:3X3:1X1:SAME"],
  ["CONV:1X1:1X1:SAME","CONV:5X5:1X1:SAME"],
  ["MAX_POOLING:3X3:1X1:SAME","CONV:1X1:1X1:SAME"]
],
"DepthContact"].
```

It's interesting that CNN model can be described by such simple style. But we just have layer operations without input, output and other operations like skip operation. Skip operation is in ResNet which is very important for its architecture. Here we can use some special file like JSON and XML to define model extending string sequence's shortage.

3.2 Generate Model Representation

Next we can generate different model by generating different string sequences with recurrent neural network. RNN is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. So, CNN architecture can generate by RNN family.

In AutoML, an RNN controller is trained in a loop: the controller first samples a candidate architecture, i.e. a child model (String sequence), and then trains it to convergence to measure its performance on the task of desire which we will talk in the next section. The controller then uses the performance as a guiding signal to find more promising architectures.

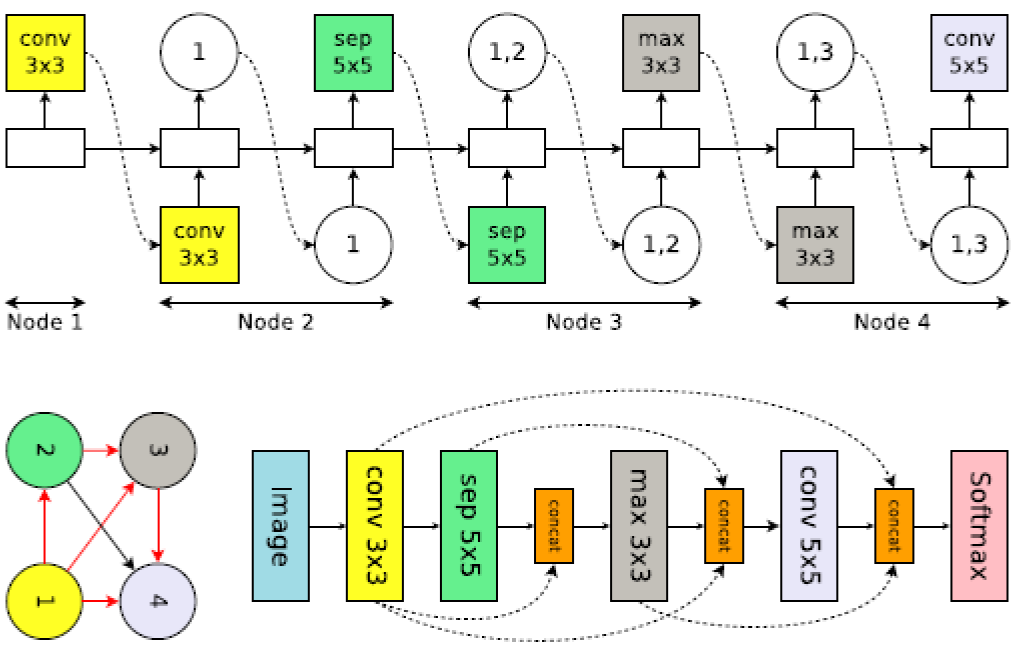

Figure.5 Example

Figure 5. shows an example run of a recurrent cell in our search space with 4 computational nodes, which represent 4 layers in a convolutional network. Top: The output of the controller RNN. Bottom Left: The computational DAG corresponding to the network's architecture.

Red arrows denote the active computational paths. Bottom Right: The complete network. Dotted arrows denote skip connections. The controller RNN samples two decisions at each decision block: 1) what previous node to connect to and 2) what activation function to use. In the search space for convolutional models, the controller RNN also samples two sets of decisions at each decision block: 1) what previous nodes to connect to and 2) what computation operation to use. These decisions construct a layer in the convolutional model. We don't talk about details here.

3.3 Neural Architecture Search

There are two aspects to improve model performance, through increasing neural network search space and generating better model. Increasing neural network model means that we have more model representation to search. Generating better model means we can generate model literately and better than previous version.

How can we increase neural network search space? Design convolutional cells and design operation between layers. Rather than designing the entire convolutional network, one can design smaller modules and then connect them together to form a network. Figure 6 illustrates this design, where the convolutional cell and reduction cell architectures are to be designed.

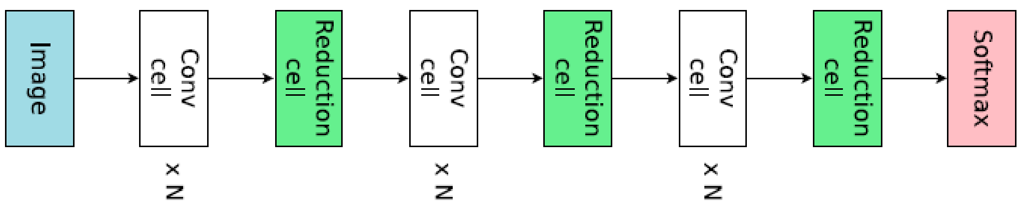

Figure 6. Connecting 3 blocks, each with N convolution cells and 1 reduction cell, to make the final network.

AutoML utilize the computational DAG with B nodes to represent the computations that happen locally in a cell. In this DAG, node 1 and node 2 are treated as the cell's inputs, which are the outputs of the two previous cells in the final network (see Figure 6). For each of the remaining B − 2 nodes, they ask the controller RNN to make two sets of decisions: 1) two previous nodes to be used as inputs to the current node and 2) two operations to apply to the two sampled nodes. The 5 available operations are: identity, separable convolution with kernel size 3×3 and 5×5, and average pooling and max pooling with kernel size 3×3. At each node, after the previous nodes and their corresponding operations are sampled, the operations are applied on the previous nodes, and their results are added.

How can we improve expected reward literately? Google tested many methods like grid search, reinforcement learning, stochastic search, Bayesian search and so on. Here I just introduce reinforcement learning.

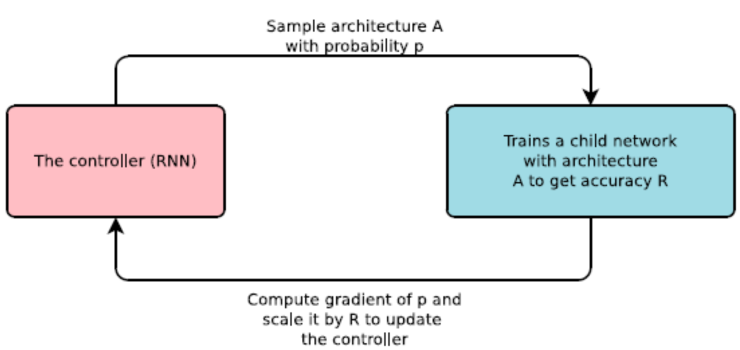

Figure 7. An overview of neural architecture search with RL

The agent sequentially selects layers via the ε-greedy strategy until it reaches a termination state(See Figure 7). The CNN architecture defined by the agent's path is trained on the chosen learning problem, and the agent is given a reward equal to the validation accuracy. The validation accuracy and architecture description are stored in a replay memory, and experiences are sampled periodically from the replay memory to update Q-values. The agent follows an ε schedule which determines its shift from exploration to exploitation. Their method requires three main design choices: 1) reducing CNN layer definitions to simple state tuples, 2) defining a set of actions the agent may take, i.e., the set of layers the agent may pick next given its current state, and 3) balancing the size of the state-action space—and correspondingly, the model capacity—with the amount of exploration needed by the agent to converge. We do not talk details here.

4 Conclusions

Since I don't have much time to do detailed work, this report is a little simple and just summarizing the methods of automated deep learning. AutoML is the first automating machine learning system that did a lot of very instructive research. So I mainly introduce their work about how to automated search good convolutional neural network through two aspects: increasing neural architecture search space and improving model performance. First we can design unique CNN cells and create different block connections, second we can improve expected reward by reinforcement learning.

Everyone can do deep learning when deep learning can be automated done. It's so amazing that I am attracted by this topic, and also did some work that cannot write down here. This filed is still have a long way to go and I think also it will be very promising in the future.

Acknowledgments

I want to thank my girlfriend Yu, my mentor Zhongwei Wang for their help and suggestions when I do this report.

References

[1] Baker, Bowen, Gupta, Otkrist, Naik, Nikhil, and Raskar, Ramesh. Designing neural network architectures using reinforcement learning. In ICLR, 2017a

[2] Baker, Bowen, Otkrist, Gupta, Raskar, Ramesh, and Naik, Nikhil. Accelerating neural architecture search using performance prediction. *Arxiv, 1705.10823*, 2017b

[3] Bello, Irwan, Pham, Hieu, Le, Quoc V., Norouzi, Mohammad, and Bengio, Samy. Neural combinatorial optimization with reinforcement learning. In ICLR Workshop, 2017a.

[4] Brock, Andrew, Lim, Theodore, Ritchie, James M., and Weston, Nick. SMASH: one-shot model architecture search through hypernetworks. ICLR, 2018.

[5] Cai, Han, Chen, Tianyao, Zhang, Weinan, Yu, Yong., and Wang, Jun. Efficient architecture search by network transformation. In AAAI, 2018.

[6] Chollet, Francois. Xception: Deep learning with depthwise separable convolutions. In CVPR, 2017.

[7] Deng, Boyang, Yan, Junjie, and Lin, Dahua. Peephole: Predicting network performance before training. Arxiv, 1705.10823, 2017.

[8] DeVries, Terrance and Taylor, Graham W. Improved regularization of convolutional neural networks with cutout. Arxiv, 1708.04552, 2017.

[9] James Bergstra, Daniel Yamins, and David D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. ICML (1), 28:115–123, 2013.

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

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In European Conference on Computer Vision, pp. 630–645. Springer, 2016.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[13] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. CVPR, pp. 3367–3375, 2015.

[14] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In NAACL, 2016.

[15] Marcin Andrychowicz, Misha Denil, Sergio Gomez, MatthewWHoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. arXiv preprint arXiv:1606.04474, 2016.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.