Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
L3S - Forschungszentrum
Technische Informationsbibliothek (TIB)

Relative depth estimation with RNN-based model

Laborbericht

im Studiengang Informatik

von

Eric Roslin Wete Poaka
10001113

Armel Dedjouong
3082880

Prüfer: Matthias Springstein
Zweitprüfer: Prof. Ralph Ewerth
Betreuer: Matthias Springstein

Hannover, 28. September 2017

Relative depth estimation with RNN-based model

Eric Wete
Leibniz University of Hannover

Armel Dedjouong
Leibniz University of Hannover

MSc. Matthias Springstein
Graduate Research Assistant
L3S - Research Center
Leibniz University of Hannover
Email: matthias.springstein@tib.eu

Deep learning and recurrent neural network techniques are useful in Machine Learning domains for instance Image, text, speech recognition and Image classifications. We propose in this project an approach based on residual network and LSTM architecture. The residual network we built is based on algorithms and the architecture presented in [1]. We also used metrics introduced with [2] to compare our results and discuss them. Our method estimates with good accuracy the relative depth between two points of an image input.

It came out that after 24 hours training of our network,the estimation accuracy was already more than 70 percent and it was always growing up.

1 Introduction

Recurrent neural networks are used to be very useful in many domains such as machine learning. A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This property allows it to exhibit dynamic temporal behavior. According to [3] it could be seen as a neural sequence model that achieves state of the art performance on important tasks including language modeling, speech recognition, and machine translation.
In comparison to feed-forward neural network like (e.g. Convolutional neural network: CNN), RNNs could use their own internal memory to process arbitrary sequences of inputs. Their application can be found in handwriting recognition and speech recognition.

The goal of our work is to estimate the relative depth (difference) between two points (regions).
To achieve our goal we get used depth prediction in monocular images, deep learning and recurrent neural networks approaches.

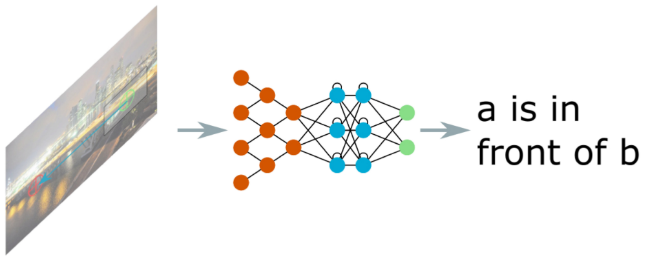

Fig. 1. Our work's goal.

2 Related Work

Residual Learning. In image classification [1], the authors ease the training of network using a residual learning framework. The layers have been reformulated as learning residual functions with the reference to layer inputs instead of learning unreferenced functions. They provided a residual network which could be easily optimized and gain accuracy from increased depth. The resulting network has a big improvements compared to other methods and proceeded to some datasets.

Learning Ordinal Relationships. For Mid-Level Vision, a framework has been realized to estimate visual properties of an image by learning about ordinal relationships [2]. From the input image, the system tried to estimate relationship between two pairs of pixel sequentially. This estimation of order relationship for pairs of points of an input image leads to accuracy over metric estimation. The analysis showed that the resulting models performed well and they learnt from simple rules to make decisions that are in order.

3 Deep Residual Learning

Our approach use residual network and Long short-term memory (LSTM) techniques.
For each input, we compute the residual network and its resulting LSTM cell. More precisely, an image is subdivided in several inputs (here 16) and each of them is an input of

the residual network.Its output served as input of the LSTM cells network.

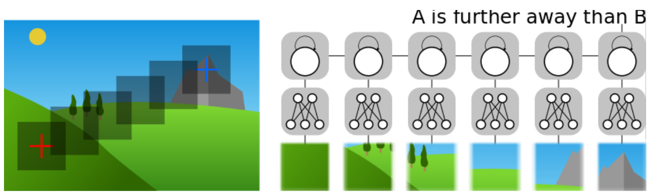

Fig. 2. Our Approach

3.1 Residual Learning

Our approach use an algorithm from [1] which has been customized. Here we realized a network architecture based on 29 Layers corresponding to the figure 3.

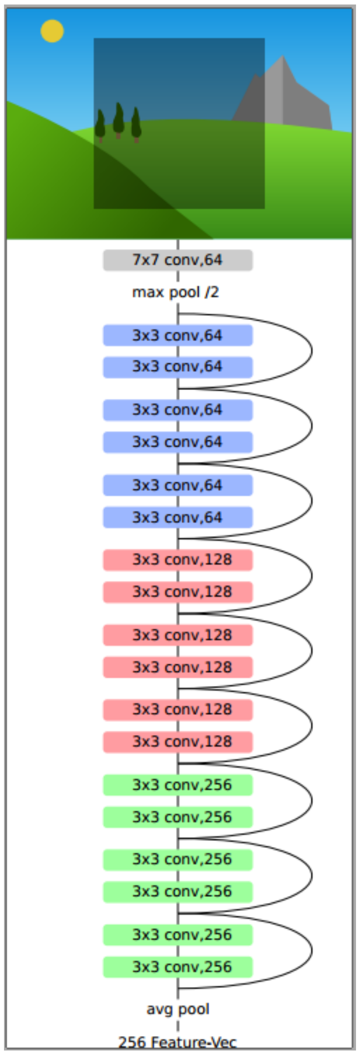

Fig. 3. Residual network architecture.

The architecture has been improved with Deeper Bottleneck Architectures. The improvement is made by applying a three layers stack for each ImageNets residual function as shown in [1].
Both stacks are shown in Figigure 4. The first layer is a 1x1 convolution.It reduces the dimensions and transfers the bottleneck with smaller dimensions to the 3x3 layer . The 3x3 layer runs and produces its result with the same input size. The last 1x1 layer restores the initial dimensions.

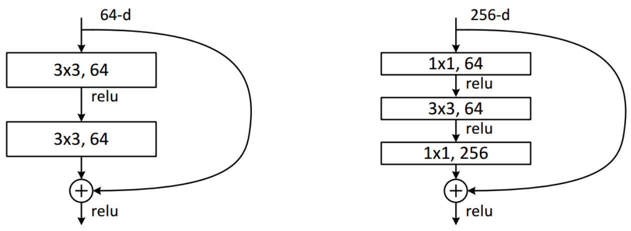

Fig. 4. A deeper residual function for ImageNet. Left: 2 Layers stack. Right: 3 Layers stack.

3.2 LSTM

Based on [4], Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture that remembers values over arbitrary intervals. This architecture suits for classification, process and time series prediction. It consists of block which can be activated depending on its guards, whose values depends on the previous state of the network or the previous block.
Our approach build a network by passing the output value of the previous cell as input to the next cell. See 3.

Our methods ends with a fully connected layer which gives predicted output values i.e. which point is in foreground.

3.3 Adam Optimizer

Adam (adaptive moment estimation) as in [5] is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.
It is used in our Code to minimize the sum of softmax cross entropy between the true labels and the calculated labels.
We first set the learning rate and the initial parameters randomly. These parameters are randomly shuffled in the training set until an approximate minimum is obtained. The learning rate is considered as a step size.

3.4 Implementation

Our approach has been implemented using the library *Tensoflow*[1]. The programming language used is *Python*[2]
We get the input image files and the labels associated with it. We processed the images to have 16 sequences of image RGB with batch size of 32. Each element of the sequence is processed using the residual network. Based on the output we build a LSTM blocks chain and the last LSTM block's output is computed inside a fully connected layer. The end results correspond to the predicted labels.
At this step, we compute accuracy comparing the predictions and the labels as input. We train our network until we get a predefined accuracy percentage or on a predefined maximal number of iterations.
The graph is periodically saved during the training of the network so it could be resume afterwards.

4 Results

Our Dataset contains training and validation data. To train our network, we use several images and their corresponding labels. The training run 320700 iterations. The evolution of the training is shown in the figure 5. We can observe that the training started with a low accuracy of about 0.45. But this precision has evolved gradually. It exceeds 0.5 after about 5000 images. Towards the end, an average value of about 0.8 can be observed.

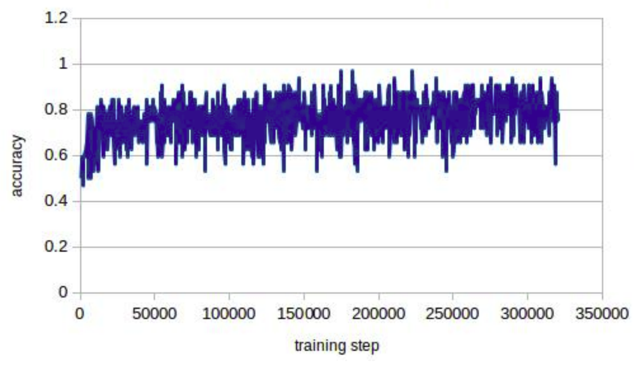

Fig. 5. Training evolution of the network

This result can be observed in more detail in table 1. To build this table, we select the values at the beginning and at the end of the training with an interval of $10,000 \pm 500$ between two consecutive values. We observe at the beginning that the accuracy is not yet acceptable because we still have low values like 0.625. But we continue to train the network until we get a more acceptable accuracy. The last values shows us that this accuracy has considerably increased.

[1]An open-source software library for Machine Intelligence. Apache 2.0-licensed Theano-like library with support for CPU, GPU and Google's proprietary TPU (tensor processing unit), mobile. `https://www.tensorflow.org`

[2]Python is a widely used high-level programming language for general-purpose programming. `https://www.python.org`

Table 1. Accuracy of training

Step (begin)	Accuracy	Step (end)	Accuracy
10300	0.562	230400	0.812
20800	0.687	240500	0.843
30800	0.812	250400	0.843
40900	0.750	260800	0.781
50800	0.781	270800	0.843
60900	0.750	280700	0.718
70900	0.812	290600	0.875
80900	0.812	300800	0.718
90900	0.750	310800	0.812
100900	0.625	320700	0.750

It has an average value of 0.8055 and a minimum value of 0.75 which is acceptable.

After the training, we use the second data to validate our network. The validation runs for 320000 iterations.

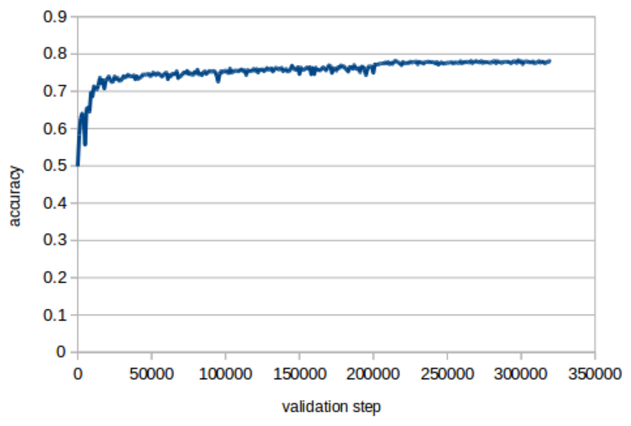

Fig. 6. validation of the network

The network presents at the beginning of the validation an accuracy of about 50%. But after about 1000 steps, this precision goes above 70%. At the end, stability can be observed around 80%.

This stability is better observed in table 2 with a standard deviation of $\sigma = 0.015$ without the first accuracy. To build this table, we chose each accuracy after an interval of 50,000 values of the previous one and the last value is that of our last validation image. Looking at the figure 6 and the table 2, we can say that the results of our approach are very good.

Table 2. Accuracy of network validation

Step	Accuracy	Step	Accuracy
1	0.498	250001	0.776
50001	0.742	200001	0.750
100001	0.755	300001	0.780
150001	0.746	320001	0.779

5 Discussion

We propose a solution estimating the relative depth between two points of an image based on recurrent neural networks and LSTM techniques. The network realized gives good result with a high value of accuracy showing that our network is performing well. Since the training may take a while we decide to have checkpoint to save the network state periodically to be able to resume afterwards. Future work could be to allow more layer (output) for the recurrent network or to improve our network architecture with another level of LSTM in the current architecture.

References

[1] Kaimin He, Xiangyu Zhang, S. R., and Sun, J., 2015. "Deep residual learning for image recognition". pp. 1–8.

[2] Daniel Zoran, Phillip Isola, D. K., and Freeman, W. T. "Learning ordinal relationships for mid-level vision". pp. 1–8.

[3] Zaremba, W., Sutskever, I., and Vinyals, O., 2014. "Recurrent neural network regularization". *CoRR,* **abs/1409.2329**.

[4] Hochreiter, S., and Schmidhuber, J., 1997. "Long short-term memory". pp. 1735–80.

[5] Kingma, D. P., and Ba, J., 2014. "Adam: A method for stochastic optimization". *CoRR,* **abs/1412.6980**.