Research Plan/Project Summary

Dense face detection and improving temporal convolutional networks for automatic image captioning

Nikita Zozoulenko

April 18, 2018

1 Rationale

My work is going to be split into the following three parts: (i) A derivation of the general case for a convolutional neural network (CNN) with an arbitrary input of a tensor of order 4 to be used for my own neural network library implementation, (ii) the systematic construction and evaluation of a fully convolutional model for detecting a variable number of faces for real time video, and (iii) creating and improving temporal convolutional networks (TCN) to reach a new state of the art, and to then apply the model on a more complex task which TCNs have never been used for before, namely automatic image captioning, to try to outperform traditional LSTMs and GRUs.

1.1 Derivation of a CNN

When i first stated out in the field of deep learning, I found that the field lacked fully derived examples of convolutional neural networks. Since the model uses tensors of order 4 to represent activations, they cannot be visualized in an effective manner. This causes educational sources to only explain the simple two-dimensional case, instead of the four-dimensional case which is required for the network to perform well. For instance, once concept which revolutionized the training of CNNs is Batch Normalization, and requires all dimensions (a tensor of order 4) to function. It is therefore of the utmost importance that education resources show derivations of the order 4 case, something which no educational resource does. Only teaching students the simple case of order 2 (batch size of one, one channel, and two spatial dimensions), hinders them from reaching their full potential regarding applying CNNs to a variety of problems.

The aim is to present a clear derivation of the general case for convolutional neural networks with an arbitrary input of a tensor of order 4, including varying strides and zero-padding. The aim is to then use my derivations to implement feed-forward and convolutional neural networks with only elementary math operations in Python and C++ for my own library, to gain an deeper understanding and intuition of neural networks. I want to validate my own implementation by training the models on a basic task of classifying handwritten digits, a task where models can be easily trained on a CPU without GPU-acceleration.

1.2 Dense Face Detection

Conventional face tracking and detection methods are limited to a few or just a single face. The aim of this paper is additionally to construct a dense face detector using a fully convolutional neural network which will be capable of detecting a variable number of faces for real time video. This method should be able to be applied to CCTV or other security systems, or areas such as person tagging on social media and face detection for digital cameras.

There are multiple moderns techniques and network architectures used for multi-class object detection. The aim of this paper is to systematically and empirically investigate how the most successful and popular methods, such as Feature Pyramid Networks, Focal Loss, Online Hard Example Mining, and data augmentation affects the performance of a single class object detector. Since object detectors often are used for specifically multi-class classification and detection, the field lacks structured data and empirical evaluations on how these methods affect binary classification problems (e.g. face or no face).

1.3 Improving TCNs for Image Captioning

Additionally, the purpose of this paper is to improve temporal convolutional networks (TCN) in two areas which I have found lacking in the original authors implementation. Firstly, their Pytorch implementation of TCNs is slightly computationally inefficient due to their zero-padding scheme. I'm going to propose my zero-padding method, which final result is mathematically equivalent to their implementation, but is slightly more computationally efficient, while still making sure there's no information leakage from the future into the past. Secondly, I will propose a new dilation scheme which takes inspiration from how conventional spatial state of the art CNNs function. These changes should increase both the efficiency and accuracy of the models, and I will empirically evaluate them on two task. Furthermore, I will compare three different residual building blocks to see their effect on the performance and accuracy of the model.

Improving this sequence model, which has recently been found to perform better than its recurrent counterparts, has an effect on every sequential task. Showing empirical evidence of how TCNs perform better than GRUs and LSTMs, which are traditionally used for sequential tasks, lies in everyone's best interest to further the state of the art in sequential modeling. This can lead to the eventuality of replacing every recurrent network with its temporal convolutional counterpart, while increasing the accuracy on any task.

This is why I'm going to further the research into TCNs by applying them to a more complex task which TCNs have never been applied to before, namely automatic image captioning. By showing that TCNs also outperform GRUs and LSTMs on more complex tasks, a full transition from traditional recurrent network to temporal convolutional networks can be made.

2 Research Questions

How is a convolutional neural network derived with an arbitrary input of a tensor of order 4, with arbitrary strides and zero-padding?

How do factors such as Online Hard Example Mining, Focal Loss, random color jitter data augmentation, and Feature Pyramid Networks empirically affect the accuracy of a fully convolutional dense face detector.

How can temporal neural networks be improved in regards to computational efficiency and accuracy?

How do temporal convolutional networks compare to Gated Recurrent Units on the task of automatic image captioning?

3 Hypothesis / Engineering goals and Expected Outcome

3.1 Derivation of a CNN

The goal is to present a clear derivation of the general case for a CNN with an arbitrary input of a tensor of order 4. The derivation should be based on first principles, using the definition of what a discrete convolution does, and expanding it for a varying batch size, number of channels, width, and height. The derivation should be clear to follow and should serve as a baseline for anyone who wants to it as an educational resource to understand the underlying mathematics behind

convolutional neural networks. It is then to be used for my own open-source implementation of feed-forward and convolutional neural networks.

3.2 Dense Face Detection

The engineering goal is to construct a face detector which is capable of detecting up to hundreds of faces in an image or video, in real time. The model should satisfy three ciretia: 1. Be able to run more than 30 times a second, 2. Be able to detect faces of widely different scales, and 3. Be fully convolutional. The face detector should be able to serve as a baseline for any system which incorporates surveillance, security, or the detection of humans (e.g. tagging on social media or face focusing for digital cameras), at a variety of scales and sizes.

In addition to the engineering goals, this work will serve as research into how modern computer vision techniques affect the performance of the model. Factors which will be systematically tested and evaluated will be Online Hard Example Mining (OHEM), Focal Loss, binary cross entropy, random color jitter data augmentation, the addition of a Feature Pyramid Network (FPN) architecture, and increased network depth.

All of these factors, OHEM, FPN, focal loss, network depth, has been shown to increase the accuracy of multi-class object detectors. Because of this, I hypothesize that they also will increase the accuracy of my single-class object detector. Increased network depth for ResNets have shown to increase classification accuracy. OHEM and focal loss were designed to improve negative to positive class imbalance, which will be of importance to my face detector. FPNs have shown to create feature maps of higher quality for object detectors, which should increase the accuracy of my single-class object detector.

3.3 Improving TCNs for Image Captioning

For my first improvement to TCNs, I hypothesize that using a linear instead of an exponential dilation increase will increase the accuracy of the model. In the original implementation with an exponential dilation scheme, every input neuron's value is used exactly once to predict the final output neuron for an arbitrary timestep. This is can be compared to using the same value for a layers kernel size and stride in a conventional spatial CNN, which the best performing models doesn't use. Using linear dilation increases will enable the receptive field of neurons to overlap, similar to normal CNNs.

Additionally, a too big of an increase of dilation causes the last layers with kernel size $k \leq 2$ to effectively turn into convolutions with kernel size $k = 1$. This is due to that every kernel weight, except those at the same timestep, is being multiplied by a zero-padded neuron. The rest of the kernel will be placed on intervals of d timesteps earlier, where d is the dilation. If d is large enough, the next activation used in the convolution will always be a zero-padded neuron, and thus the whole layer effectively becomes a convolution with kernel size $k = 1$. The effect of exponential dilations in deep architectures is that only the first layers use data from previous timesteps, while later layers simply become mathematically equivalent to normal feed-foward layers across a single timestep.

The second improvement is to use a more efficient zero-padding scheme which is mathematically equivalent to the original implementation, but is slightly more computationally efficient. The implementation details are detailed in the next section.

For the task of automatic image captioning, I hypothesize that the temporal convolutional model will perform better than its recurrent counterpart, a Gated Recurrent Unit (GRU). This is based

on the fact that TCNs have been shown to perform better than recurrent models (GRUs, LSTMs and vanilla RNNs), on a variety of more simple tasks.

4 Procedures and Data Analysis

4.1 Derivation of a CNN

I will derive the convolutional neural network based on first principles, based on the definition of forward propagation, backpropagation and a discrete convolution, and expand it to fit an arbitrary input of a tensor of order 4. This will be done with pen and paper and then transferred over digitally to LaTeX. The derived partial derivatives required to compute the gradient of the loss function will be approximated and compared to its numerical approximated using the formal definition of a partial derivative.

For my own library implementation of feed-forward and convolutional neural networks, I will use the linear algebra library Eigen for matrix operations in C++, and the scientific computing library NumPy for tensor operations in Python. The Eigen implementation will use the *MatrixXd* class to represent the neurons and weights. In the Python implementation, the NumPy *ndarray* class will be used to express tensors. The forward propagation, backpropagation and stochastic gradient descent algorithms will be implemented as described by my mathematical derivation of the general case.

To evaluate my implementation, I will train a feed-forward neural network and a convolutional neural network on the task of classifying handwritten digits using the MNIST dataset.

4.2 Dense Face Detection

For the task of detecting a variable number of faces for real time video, I will systematically investigate how focal loss, binary cross entropy, online hard example mining, feature pyramid networks, color jitter, anchors settings, and network depths affects model performance. This will be done by starting with a baseline model, using a ResNet-18, extracting feature tensors at different depths of the network, to enable the detection of faces at different scales. These features will be given as an input to two sub-networks, a classification head and a regression head. The classification head predicts the probability that certain baseline bounding boxes scattered across the image, called anchor boxes, contains a face. The regression head predicts coordinate offsets for all 4 coordinates of the anchor box, to more closely match the ground truth. During training, every predicted bounding box will be given either a negative or positive label, and a loss function will be constructed as a function of the positive and negative bounding boxes, using a combination of a cross entropy classification loss and a smooth L1 regression loss. The GPU-accelerated dynamic tensor library PyTorch will be used to enable the model to be efficient enough to run in real time.

The model will be trained and evaluated on the WIDERFace dataset. The evaluation metric used will be mean average precision (mAP). It is calculated by plotting the accuracy of the model as a function of its recall. The mean average precision is given by the integral of accuracy with respect to the recall. The model will be evaluted on the easy, medium and hard validation splits, given by the dataset authors.

All training will take place on a GPU I bought exclusively for training machine learning models, an Nvidia GTX 1080 ti, on my personal computer. Since my computational resources will be limited to a single GPU, I cannot test all combinations of factors one by one, which would

require training a minimum of $2^6 = 64$ unique models. Therefore, I will use my baseline model and change one architecture design decision at a time, iteratively improving the model.

4.3 Improving TCNs for Image Captioning

4.3.1 Improving TCNs

I will implement an open-source model of my improved TCN in PyTorch, based on my two suggested improvements. The network architecture hyperparameters and structure will be picked to be as similar to the original model as possible, only varying the dilations, such that the total receptive field of the output neurons is slightly bigger than the maximum sequence size. In addition to using the same residual block presented in the original implementation, I will experiment on two additional residual blocks to see their effect on the accuracy of the model.

The models will be evaluated on the SequenceMNIST and Permuted SequenceMNIST tasks.

4.3.2 Automatic Image Captioning

For the task of automatic image captioning, I will compare a TCN with a traditionally used Gated Recurrent Unit (GRU) on the the MSCOCO Captions dataset.

The models will generate image captions by sequentially suggesting the next word in a sentence, from a large vocabulary of allowed words to choose from. The vocabulary will be chosen based on the most used words in the training data. Words in the training image captions not included in the vocabulary were replaced with an unknown token: $<UKN>$. The vocabulary also includes an end of string token, $<EOS>$, used by the models to signal that the generated sentence is complete. Each word will be represented as a one-hot vector encoding, where every element but one is set to zero, and the index of the non-zero element determines what word it is.

During training, each word will be given sequentially to the model, and the model will predict the next word in the sentence. It will do this by predicting a vector with N values, where N is the vocabulary size. The vector will then be put into a softmax function, and its elements interpreted as the probability that the selected word should be the next word in the sentence. The model will not ignore the $<UKN>$ tokens, and will train to predict them the same as any other word. At test time if $<UKN>$ were to be picked, it will be replaced by the word prediction with the second highest probability.

A ResNet will be used to extract features from the input images, which will be incorporated by the sequential model to base their predictions off. The GRU will initialize its first hidden layer with the features from the image extractor. Both models will use the extracted image feature vector as a start of string token, at timestep $t = 1$.

The models will be evaluated using the MSCOCO Captions evaluation server on the publically unavailable test set, on five different automated evaluation metrics: CIDEr-D, METEOR, Rogue-L, BLUE and SPICE.

5 Risk and Safety

No special risk and safety precautions have to be taken. Data used will be taken from a publicly available datasets and will not be used for anything other than training the machine learning model. I will not for example distribute or use the data for any other means.

References

[1] *Microsoft COCO Captions: Data Collection and Evaluation Server.* X. Chen and H. Fang and TY Lin and R. Vedantam and S. Gupta and P. Dollár and C. L. Zitnick. arXiv preprint arXiv:1504.00325, 2015

[2] *Training Region-based Object Detectors with Online Hard Example Mining.* A. Shrivastava, A. Gupta, and R. Girshick. arXiv preprint arXiv:1604.03540, 2017

[3] *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.* B. Shaojie, K. Zico, and K. Vladlen arXiv preprint arXiv:1803.01271, 2018.

[4] *Focal Loss for Dense Object Detection.* Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He and Piotr Dollár. arXiv preprint arXiv:1708.02002, 2017

[5] *Feature Pyramid Networks for Object Detection.* Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan and Serge J. Belongie. arXiv preprint arXiv:1612.03144, 2016

[6] *Deep residual learning for image recognition.* K. He, X. Zhang, S. Ren, and J. Sun. arXiv preprint arXiv:1512.03385, 2015.

[7] *The MNIST database of handwritten digits* Y. LeCun, C. Cortes and C. Burges. Courant Institute, NYU. Google Labs, New York. Microsoft Research, Redmond. URL http://yann.lecun.com/exdb/mnist/. 3 November 2017.

[8] *WIDER FACE: A Face Detection Benchmark.* Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016

[9] *Automatic differentiation in PyTorch.* A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer 2017

[10] *CS231n: Convolutional Neural Networks for Visual Recognition.* F. Li, A. Karpathy and J. Johnson. Stanford University, university course, winter 2016.

[11] *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* S. Ioffe and C. Szegedy. arXiv preprint arXiv:1502.03167, 2015.

[12] *Eigen v3.* G. Guennebaud and B. Jacob and others. URL http://eigen.tuxfamily.org. 2010

[13] *The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30.* S. van der Walt, S. C. Colbert and G. Varoquaux. (2011), DOI:10.1109/MCSE.2011.37