Bsc notes

André O. Andersen

February 10, 2021

Newell et al. - Stacked Hourglass Networks for Human Pose Estimation

- We introduce a novel "stacked hourglass" network design for predicting human pose. We refer to the design as an hourglass bsed on our visualization of the steps of pooling and subsequent upsamling used to get the final output of the network. The hourglass network pools down to a very low resolution, then upsamples and combines features across multiple resolutions.
- We expand on a single hourglass by consecutively placing multiple hourglass modules together end-to-end. This allows for repeated bottom-up, top-down inference across scales. In conjunction with the use of intermediate supervision, repeated bidirectional inference is critical to the network's final performance.
- Our hourglass module before stacking is closely connected to fully convolutional networks. Our hourglass module differs from these designs mainly in its more symmetric distribution of capacity between bottom-up processing (from high resoltuions to low resolutions) and top-down processing (from low resolutions to high resolutions)
- The hourglass is set up as follows: Convolutional and max pooling layers are used to process features down to a very low resolution. At each max pooling step, the network branches off and appies more convolutions at the original pre-pooled resolution. After reaching the lowest resolution, the network begins the top-down sequence of upsamping and combination of features across scales. To bring together information across two adjacent resolutions, we do nearest neighbor upsampling of the lower resolution followed by an elementwise addition of the two sets of features. The topology of the hourglass is symmetric, so for every layer present on the way down there is a corresponding layer going up. After reaching the output resolution of the network, two consecutive rounds of $1x1$ convolutions are applied to produce the final network predictions. The output of the network is a set of heatmaps where for a given heatmap the network predicts the probability of a joint's presence at each and every pixel.
- Operating at the full input resolution of $256x256$ requires a significant amount of GPU memory, so the highest resolution of the hourglass (and thus the final output resolution) is $64x64$, This does not affect the network's ability to produce precise joint predictions. The full network starts with a $7x7$ convolutional layer with stride 2, followed by a residual module and a round of max pooling to bring the resolution down from 256 to 64.
- We take our network architecture further by stacking multiple hourglasses end-to-end, feeding the output of one as input into the next. This provides the network with a mechanism for repeated bottom-up, top-down inference allowing for reevaluation of initial estimates and features across the whole image.
- There are often multiple people visible in a given input image, but without a graphical modle or other postprocessing step the image must convey all necessary information for the network to determine which person deserves the annotation. We deal with this by training the network to exclusively annotate the person in the direct center.

poti et al. - Everything you wanted to know about Deep Learning for Computer Vision but were afraid to ask

Abstract

- This paper has the objective to introduce the most fundamental concepts of Deep Learning for Computer vision in particular CNNs, AEs and GANs, including architectures, inner workings and optimization.

Convolutional Neural Networks

A. Convolutional Layer

- A layer is composed of a set of filters, each to be applied to the entire input vector. Each fliter is nothing but a matrix $k \times k$ of weights (or values) $\mathbf{w}_i$. Each weight is a parameter of the model to be learned.

- Each filter will produce what can be seen as an affine transformation of the inpuit. Another view is that each filter produces a linear combination of all pixel values in a neighbourhood defined by the size of the filter. Each region that the filter processes is called local receptive field: an output value (pixel) is a combination of the input pixels in this local receptive field. In a convolutional layer, an output value $f(i, x, y)$ is based on a filter i and local data coming from the previous layer centered at a position (x, y).

- The most commonly used filter sizes are $5 \times 5 \times d$, $3 \times 3 \times d$ and $1 \times 1 \times d$, where d is the depth of the tensor.

- It is important to mention that the convolutional operator can have different strides, which defines the step taken between each local filtering. The default is 1, in this case all pixeels are considered in the convolution. For example, with stride 2, every odd pixel is processed, skipping the others. It is common to use an arbitrary value os stride in order to reduce the running time.

b. Activation Function

- ReLU is often use in CNNs after convolutional ayers or fully connected layers, but can also be employed before layers in a pre-activation setting.

- Activation functions are not useful after Pool layers because such layers only downsamples the input data.

C. Feature or acitvation map

- Each convolutional neuron produces a new vector that passes through the activation function and it is then called a feature map. Those maps are stacked, forming a tensor that will be offered as input to the next layer

D. Pooling

- Oftne applied after a few convolutional layers, it downsamples the image in order to reduce the spatial dimensionality of the vector.

E. Normalization

- It is common to apply normalization to both the input data and after convolutional layers.

- In input data preprocessing it is common to apply a z-score normalization, which can be seen as a whitening process.

- For layer normalization there are different approaches such as the channel-wise layer normalization, that normalizes the vector at each spatial location in the input map, either within the same feature map or across consecutive channels/maps, using L1-norm, L2-norm or variation. Other methods are Local Response Normalization or Batch normalization.

F. Fully Connected Layer

- After many convolutional layers, it is common to include fully connected layers that work in a way similar to a hidden layer in order to learn weights to classify the representation. It takes as input the reshaped (flatten) version of the data coming from the last layer

- The last layer of a CNN is often the one that outputs thE class membership probabilities of each class using logistic regression.

G. CNN architecture and its parameters

- Typical CNNs are organized using blocks of convolutional layers followed by an activation function, eventually pooling and then a series of fully connected layers which are also followed by activation functions. Normalization of data before each layer can also be applied.

H. Loss Function

- In order to avoid ambiguity of solution, it is possible to add a new term that penalizes undesired situations, which is called regularization. The most common regularization is the L2-norm.

I. Optimization Algorithms

- *Stochastic Gradient Descent*: One possible solution to accelerate the process is to use approximate methods that goes through the data in samples composed of rando mexamples drawn from the original dataset, It is common to use mini-batches. By perfoming enough iteration, we assume it is possible to approximate the Gradient Descent method. In fact, SGD is a rough approximation, producing a non-smooth convergence. Because of that, variants where proposed to compensate for that, such as AdaGrad, AdaDelta and Adam. Those variants basically use the ideas of momentum and normalization, as we describe below.

- *Momentum*: Adds a new variable α to control the change in the parameters W. It creates a momentum that prevents the new parameters W_{t+1} from deviating too much from the previous direction.

J. Tricks for Training CNNs

- *Initialization*: Random initialization of weights is important the convergence of the network. The Gaussian distribution is often used to produce the random numbers, however, for models with more than 8 convolutional layers, the use of a fixed standard deviation was shown to hamper convergence. Therefore, when using rectifiers as activation functions it is recommented to use $\mu = 0$, $\sigma = \sqrt{\frac{2}{n_l}}$, where n_l is the number of connections of a response of a given layer l; as well as initializing all bias parameters to 0

- *Minibatch size*: It can be an advantage to choose the batch size so that it fully occupies the GPU memory and choose the largest experimentally found step size. A recent paper used a linear scaling rule for adjusting learning rates as afunction of minibatch size, also adding a warmup scheme with large step-sizes in the first few epochs to avoid optimization problems.

- *Dropout*: During the forward pass of the network training stage, randomly deactivate neurons of a layer with some probability p. This method became known as a form of regularization that prevents the network to overfit.

- *Batch normalization*; Also used as a regularizer, it normalizes the layer activations at each batch of input data by maintaining the mean activation close to 0 and the activation standard deviation close to 1, and using parameters γ and β to produce a lienar transofrmation of the normalized vector. BN became a standard in the recent years, often replacing the use of both regularization and dropout.

- *Pre-processing*: the inptu data can be pre-processed in several ways:

1. Compute the average image for the whole training data and subtracting it from each image
2. z-score normalization
3. PCA whitening that first tries to decorrelate the data by projecting zero-centered original data into eigenbasis, and the takes the data in the eigenbasis and divides every dimension by the eigenvalue to normalize the scale

Bulat et al. - Human pose estimation via Convolutional Part Heatmap Regression

Abstract

- This paper is on human pose estimation using CNN. Our main contribution is a CNN cascaded architecture specifically designed for learning part relationships and spatial context , and robustly inferring pose even for the case of severe part occlusions. To this end, we propose a detection-followed-by-regression CNN cascade. The first part of our cascade outputs part detection heatmaps and the second part performs regression on these heatmaps.

1 Introduction

- For the case of non-visible parts though, learning the complex mapping from occluded part appearances to part locations is hard and the network has to rely on contextual information to infer the occluded parts' location. In this paper, we show how to circumvent this problem by proposing a detection-followed-by-regression CNN cascasde for articulated human pose estimation

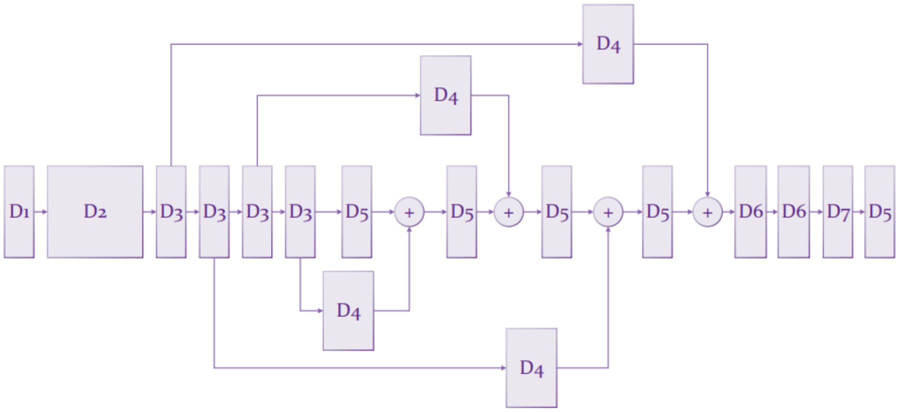
3 Method

3.1 VGG-FCN part heatmap regression

- For training on MPII, all images were cropped after centering on the person and then scaled such that a standing-up human has height 300px. All images were resized to a resolution of 380x380px. To avoid overfitting, we performed image flipping, scaling (between 0.7 and 1.3) and rotation (between -40 and 40 degrees). Both rotation and scaling were applied using a set of predefined step sizes. Augmentation were applied randomly

- The detectors were trained for about 50 epochs using a learninig rate progressively decreasing from $1e-3$ to $2.5e-5$.

- Architecture:



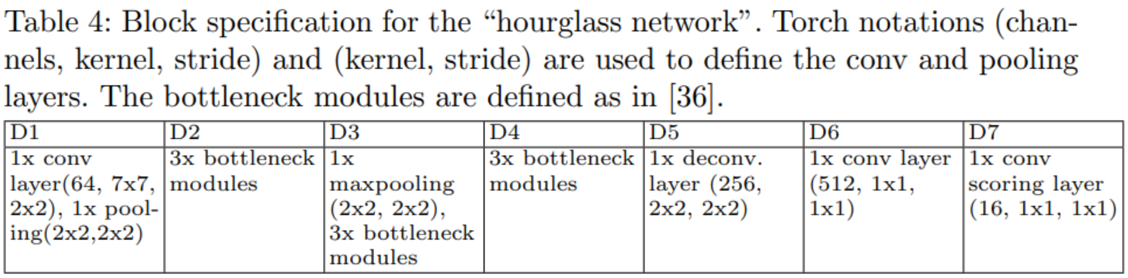


Table 4: Block specification for the "hourglass network". Torch notations (channels, kernel, stride) and (kernel, stride) are used to define the conv and pooling layers. The bottleneck modules are defined as in [36].

D1	D2	D3	D4	D5	D6	D7
1x conv layer(64, 7x7, 2x2), 1x pooling(2x2,2x2)	3x bottleneck modules	1x maxpooling (2x2, 2x2), 3x bottleneck modules	3x bottleneck modules	1x deconv. layer (256, 2x2, 2x2)	1x conv layer (512, 1x1, 1x1)	1x conv scoring layer (16, 1x1, 1x1)

- The network made us of batch normalization, and was trained with a batch size of 8.

Chu et al. - Multi-context Attention for Human Pose Estimation

3. Framework

- We adopt an 8-stack hourglass network as the baseline network. it allows for repeated bottom-up, top-down inference across scales with intermediate supervision at the end of each stack. In experiments, the input iamges are 256×256, and the output heatmaps are $P \times 64 \times 64$, where P is the number of body parts.

- We replace the residual units, which are along the side branches for combining features across multiple resolutions, by the propsed micro hourglass residual units (HRUs), and obtain a *nested hourglass network*. With this architecture, we enrich the information received by the output of each building block, which makes the whole framework more robust to scale change.

- Wothiun each hourglass, the multi-resolution attention maps are generated from features of different scales. Attention maps are then combined to generate the refined features, which are further used to generate refined attention maps and further refined features.

- Different stacks are with different semantics: lower stacks focus on local apperance, while higher stacks encode global representations. Hence attention maps generated from different stacks also encode various semantic meanings. Deeper stacks with global representations are able to recover occlusions.

Yang et al. - Learning Feature Pyramids for Human Pose Estimation

1. Introduction

- To enhance the robustness of DCNNs against scale variations of visual patterns, we design a *Pyramid Residual Module* to explicitly learn convolutional filters for building feature pyramids. Given input features, the Pyramid Residual Module obtains features of different scales via subsampling with different ratios. Then convolution is used to learn filters for features in dfifferent scales. The filtered features are upsampled to the same resolution and are summed together for the following processing.

2. Related Work

- Good initalization is essential for training deep models. Hinton and Salakhutdinov adopted the layer-by-layer pretraining strategy to train a deep autoencoder. Krizhevsky et al. initalized the weight of each layer by drawing samples from a Gaussian distribution with zero mean and 0.01 standard deviation. However, it has difficulty in training very deep networks due to the instability of gradients. Xavier initialization has provided a theoretically sound estimation of the variance of weight. It assumes that the weights are initialized close to zero, hence the nonlinear activations like Sigmoid and Tanh can be regarded as linear functions. This assumption does not hold for rectifier activations. Thus He et al. proposed an initialization scheme for recitfier networks. All the above initialization methods are derived for plain networks with only one branch.

4. Training and Inference

4.1 Initializatio Multi-Branch Networks

- Initialization is essential to train very deep networks, especially for tasks of dense prediction, where Batch Normalization is less effective because of the small minibatch due to the large memory consumption of fully convolutional networks.

5. Experiments

5.1 Experiments on Human Pose Estimation

Implementation details

- The input image is 256×256 cropped from a resized image according to the annotated body position and scale. We simply use the iamge center as the body position, and estimate the body scale by the image size. Trainign data are augmneted by scaling, rotation, flipping, and adding color noise. We use a mini-batch size of 16 for 200 epochs. The learning rate is initialized as 7×10^{-4} and is dropped by 10 at the 150th and the 170th epoch.

Tang et al. - Deeply Learned Compositional Models for Human Pose Estimation

1 Introduction

- The most recent human pose estimation systems have adopted CNN as thir backbones and yielded drastic improvements on standard benchmarks. However, they are still prine to fail when there exist ambiguities caused by overlapping parts, nearby persons and clutte rbackgrounds.