Deep Learning

Chapter 6: Advanced Components of CNNs

Mina Rezaei

Department of Statistics – LMU Munich

Winter Semester 2020

LECTURE OUTLINE

Inception Modules

Skip connections

Global average pooling

IMPORTANT TYPES OF CONVOLUTIONS

In this chapter, we discuss further advanced components of CNNs:

1. Inception Modules
2. Skip Connections
3. Global Average Pooling

Inception Modules

INCEPTION MODULES

- Problem setting: how do we choose the kernel size in each layer?
- This is often an arbitrary decision.
- Solution: offer the model kernels of different sizes in each layer through which it can propagate information and let it decide, which one to use to which extent.
- Side-effect: massive parameter reduction allowing for deeper architectures.
- First proposed in [Szegedy et. al , 2014].

INCEPTION MODULES

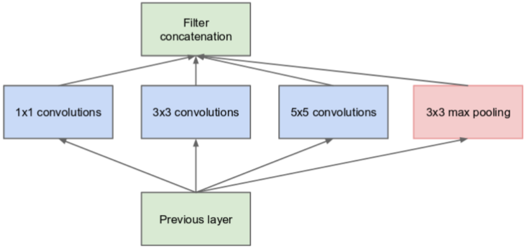
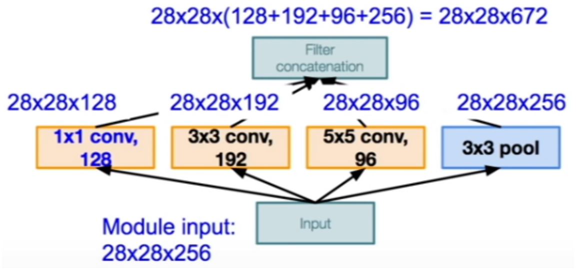


Figure: Naive inception module. The model can “choose” from kernels of different sizes.

Idea: do several convolutions in parallel and concatenate the resulting feature maps in the depth dimension. This requires equal dimensions of the feature maps created by the parallel convolutions.Thus, same padding is used throughout the parallel convolutions.

INCEPTION MODULES

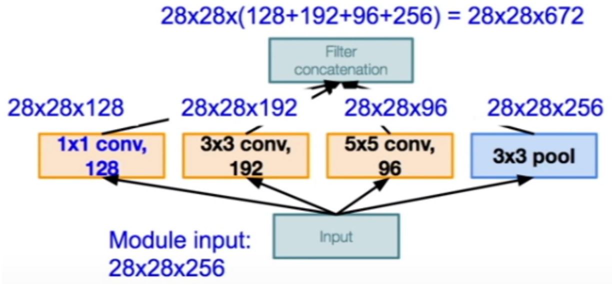


credit : Stanford University

Figure: Naive Inception module - Example

- To allow for the bypass of information throughout one inception module, an 1x1 convolutional layer is also included.
- Max-pooling is used as it is ought to increase the robustness of the feature maps. The kernels are padded accordingly to yield feature maps of equal dimensions.

INCEPTION MODULES



credit : Stanford University

Figure: Naive Inception module - Example

- Resulting feature map blocks are restricted to have the same dimensionality but can be of varying depth.
- The different feature maps are finally concatenated in the depth-dimension and fed to the next layer.

INCEPTION MODULES

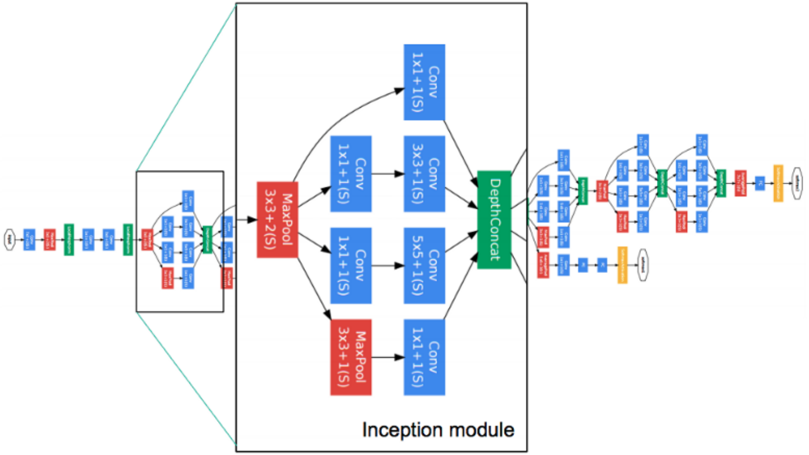


Figure: Inception modules are the integral part of the famous GoogLeNet (2014), one of the first very deep net architectures.

INCEPTION MODULES

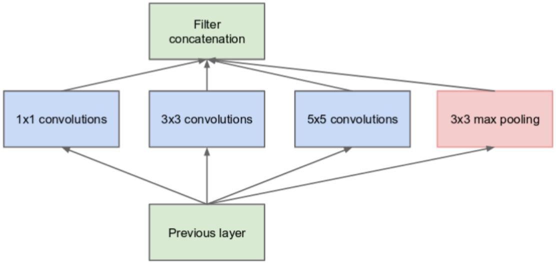


Figure: Naive inception module.

- Problem: 3x3 and 5x5 convolutions are expensive operations, especially when executed on very deep input blocks such as many feature maps from the previous layer.

INCEPTION MODULES

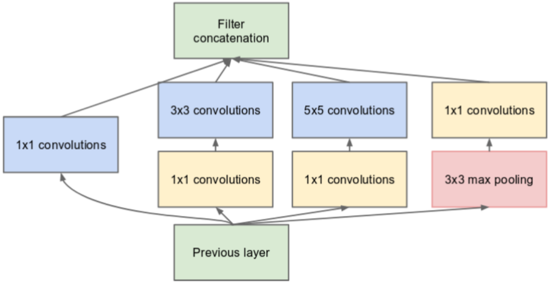


Figure: Dimensionality reduced inception module.

- Solution: apply 1x1 convolutions beforehand to reduce the depth of the previous feature map.

INCEPTION MODULES

- Let us understand this with a little numerical example.
- Output dimensions of the previous layer: [28, 28, 192].
- Output dimensions of the 5x5 convolution from the inception module: [28, 28, 32].
- The 5x5 convolution has stride 1 and same padding.
- To improve speed, we first convolve the [28, 28, 192] input with 16 1x1 kernels which results in a [28, 28, 16] block. We then apply the 32 5x5 kernel convolution on this "thinner" block.
- Required operations:
  - Naive: $5^2 \cdot 28^2 \cdot 192 \cdot 32 = 120.422.400$
  - Improved version with 1x1 convolution and depth 16: $1^2 \cdot 28^2 \cdot 192 \cdot 16 + 5^2 \cdot 28^2 \cdot 16 \cdot 32 = 12.443.648$

Skip connections

SKIP CONNECTIONS

- Problem setting: theoretically, we could build infinitely deep architectures as the net should learn to pick the beneficial layers and skip those that do not improve the performance automatically.
- But: this skipping would imply learning an identity mapping $\mathbf{x} = \mathcal{F}(\mathbf{x})$. It is very hard for a neural net to learn such a 1:1 mapping through the many non-linear activations in the architecture.
- Solution: offer the model explicitly the opportunity to skip certain layers if they are not useful.
- Introduced in [He et. al , 2015] and motivated by the observation that stacking evermore layers increases the test- as well as the train-error ($\neq$ overfitting).

SKIP CONNECTIONS

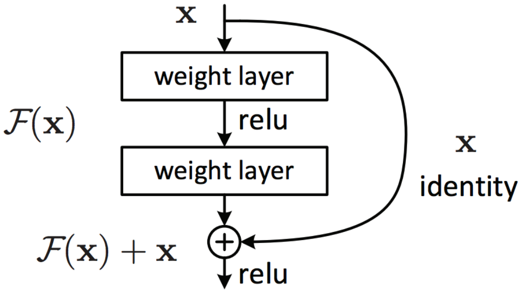


Figure: Skip connection/ residual learning module. The information flows through two layers and the identity function. Both streams of information are then element-wise summed and jointly activated.

SKIP CONNECTIONS

- Let $\mathcal{H}(\mathbf{x})$ be the optimal underlying mapping that should be learned by (parts of) the net.
- $\mathbf{x}$ is the input in layer *l* (can be raw data input or the output of a previous layer).
- $\mathcal{H}(\mathbf{x})$ is the output from layer *l*.
- Instead of fitting $\mathcal{H}(\mathbf{x})$, the net is ought to learn the residual mapping $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ whilst $\mathbf{x}$ is added via the identity mapping.
- Thus, $\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$, as formulated on the previous slide.
- The model should only learn the **residual mapping** $\mathcal{F}(\mathbf{x})$
- Thus, the procedure is also referred to as **Residual Learning**.

SKIP CONNECTIONS

- The element-wise addition of the learned residuals $\mathcal{F}(\mathbf{x})$ and the identity-mapped data $\mathbf{x}$ requires both to have the same dimensions.
- To allow for downsampling within $\mathcal{F}(\mathbf{x})$ (via pooling or valid-padded convolutions), the authors introduce a linear projection layer W_s .
- W_s ensures that $\mathbf{x}$ is brought to the same dimensionality as $\mathcal{F}(\mathbf{x})$ such that:

$$y = \mathcal{F}(\mathbf{x}) + W_s\mathbf{x},$$

- y is the output of the skip module and W_s represents the weight matrix of the linear projection (# rows of W_s = dimensionality of $\mathcal{F}(\mathbf{x})$).
- This idea applies to fully connected layers as well as to convolutional layers.

Global average pooling

GLOBAL AVERAGE POOLING

- Problem setting: tackle overfitting in the final fully connected layer.
  - Classic pooling removes spatial information and is mainly used for dimension and parameter reduction.
  - The elements of the final feature maps are connected to the output layer via a dense layer. This could require a huge number of weights increasing the danger of overfitting.
  - Example: 256 feature maps of dim 100x100 connected to 10 output neurons lead to 25.6×10^6 weights for the final dense layer.

GLOBAL AVERAGE POOLING

- Solution:
  - Average each final feature map to the element of one global average pooling (GAP) vector.
  - Do not use pooling throughout the net.
  - Example: 256 feature maps are now reduced to GAP-vector of length 256 yielding a final dense layer with 2560 weights.

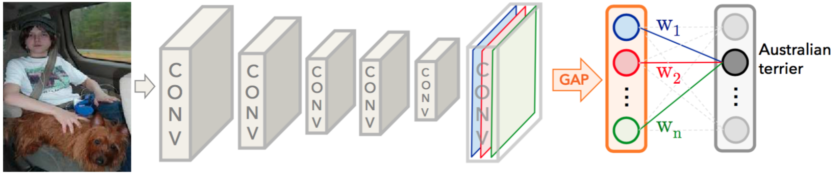


Figure: Illustration of GAP [Zhou et. al , 2016]. Each feature map representing one feature category averaged into one final vector. No pooling operations are applied throughout the net. The dimensionality of the input reduces solely due to the convolution operations.

GLOBAL AVERAGE POOLING

- GAP preserves whole information from the single feature maps whilst decreasing the dimension.
- Mitigates the possibly **destructive** effect of pooling.
- Each element of the GAP output represents the activation of a certain feature on the input data.
- Acts as an additional regularizer on the final fully connected layer.
- Allows for interpretation of the model via Class Activation Maps (more on this later).

CLASS ACTIVATION MAPPING

- We want to understand the decision-making of a net, e.g. **why does it classify image X as a cat?**
- Simplest method based on GAP was introduced in [Zhou et. al, 2016].
- Idea:
  - the final GAP vector stores the activation of each feature map category that was learnt throughout the net.
  - the dense layer that connects the output classes with the GAP vector stores information about how much each feature contributes to each class.
  - exploit this information to show which parts of the input image would be activated for each class.

CLASS ACTIVATION MAPPING

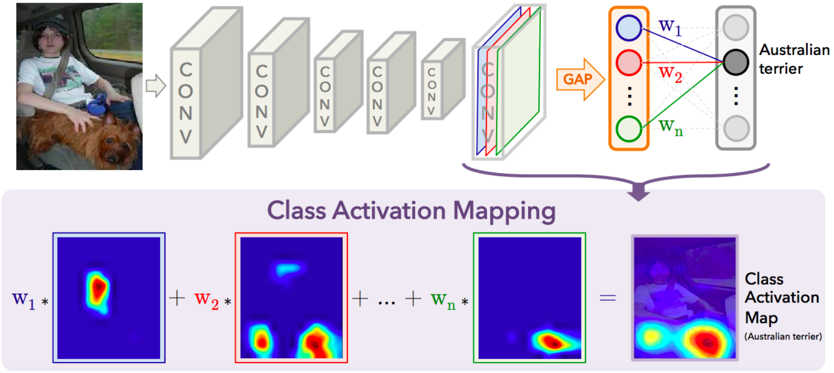


Figure: Illustration of the class activation mapping. The activated regions from the feature maps are summed up weighted by their connection strength with the final output classes and upsampled back to the dimension of the input image. No max-pooling is applied throughout the architecture, the downsampling is due to the CNN layers.

CLASS ACTIVATION MAPPING

1. Train a net with GAP pooling end-to-end.
2. Run a forward-pass with the image i you would like to understand.
3. Take the final l feature maps $f_1, ..., f_l$ for this input.
4. Get the *feature weights* $w_{j1}, ..., w_{jl}$ that connect the GAP layer with the final class output j that you would like to interpret (e.g. terrier).
5. Create the **class activation map** (CAM) for class j on input image i:

$$\text{CAM}_{j,i} = \sum_{k=1}^{l} w_{jk} * f_k$$

6. Normalize the values such that $\text{CAM}_{j,i} \in [0, 1]$.
7. In case of valid convolutions, the resulting CAM will be smaller than the input image. Linear upsampling is then used to map it back to the input dimension.
8. Overlay the input image with the CAM and interpret the activation.

REFERENCES

B. Zhou, Khosla, A., Labedriza, A., Oliva, A. and A. Torralba (2016)
Deconvolution and Checkerboard Artifacts
http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich (2014)
Going deeper with convolutions
https://arxiv.org/abs/1409.4842

Kaiming He, Zhang, Xiangyu, Ren, Shaoqing, and Jian Sun (2015)
Deep Residual Learning for Image Recognition
https://arxiv.org/abs/1512.03385

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva and Antonio Torralba (2016)
Learning Deep Features for Discriminative Localization
http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf