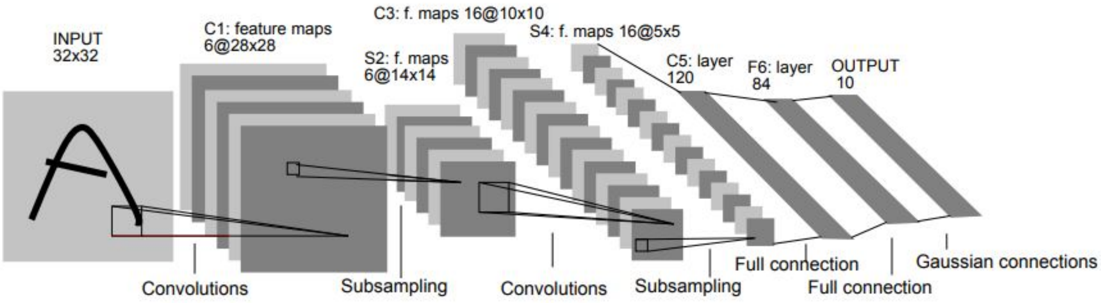
Neural Architectures: Hall of Fame

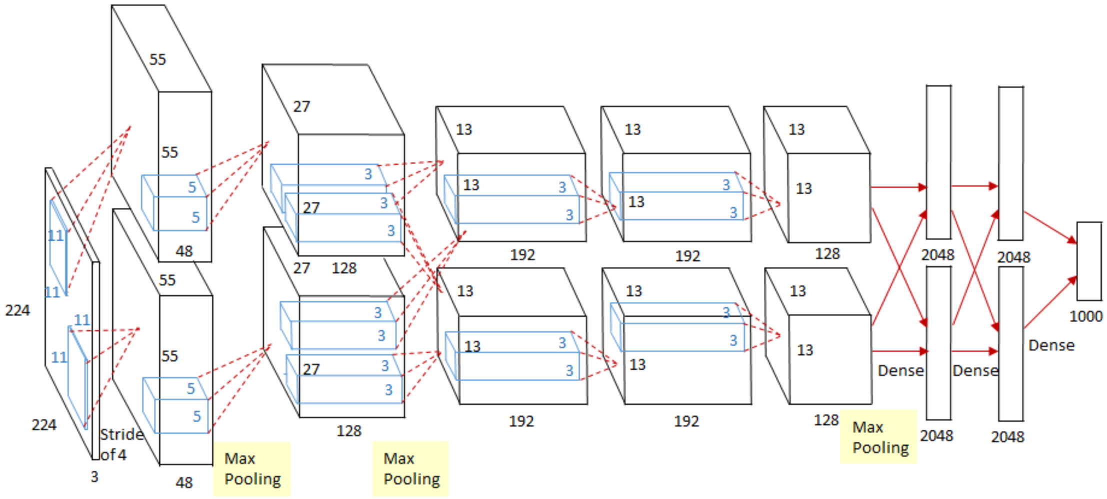
October 2, 2020

1 LeNet 5



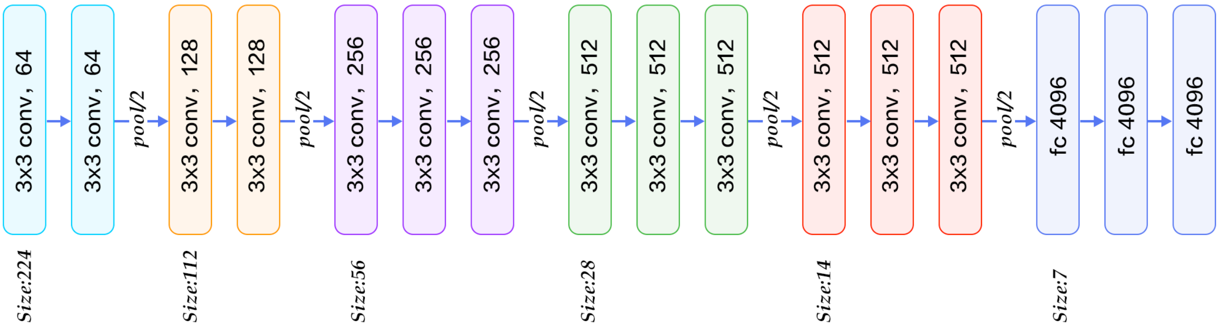
- LeNet5[1] is one of the very first convolutional neural networks.
- Convolutional neural network use sequence of 3 layers: convolution, pooling, non-linearity.
- Multi-layer neural network (MLP) as final classifier.
- Use CPUs - slow training.

2 ALEXNET



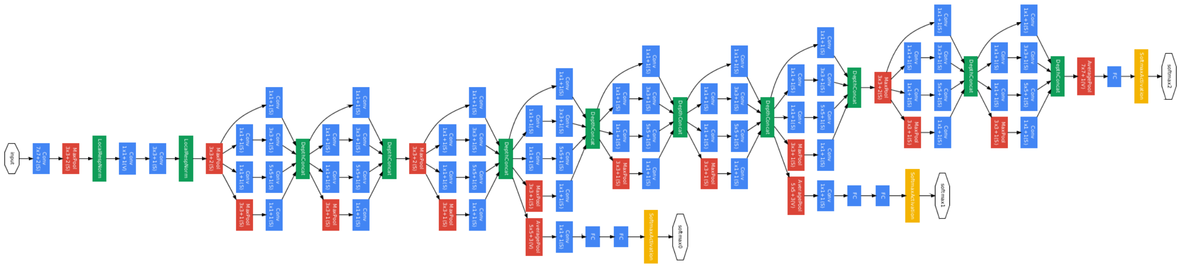
- AlexNet[2] used rectified linear units (ReLU) as non-linearities.
- Use of dropout technique to selectively ignore single neurons during training, a way to avoid overfitting of the model.
- Overlapping max pooling, avoiding the averaging effects of average pooling.
- Use of GPUs NVIDIA GTX 580 to reduce training time.

3 VGG



- VGGnet[3] consists of 16 convolutional layers with only 3 × 3 kernels.
- Smaller 3 × 3 filters in each convolutional layers.
- Multiple 3 × 3 convolution in sequence can emulate the effect of larger receptive fields, for examples 5 × 5 and 7 × 7.
- Network comprised of 13 million parameters.

4 INCEPTION



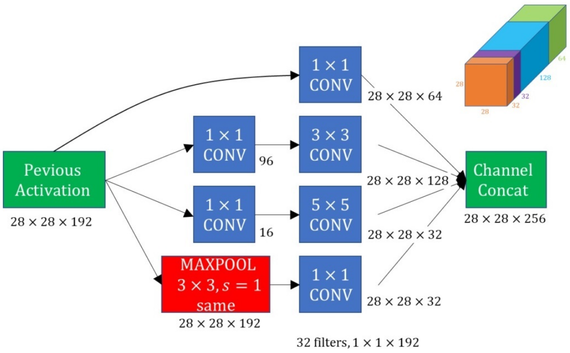
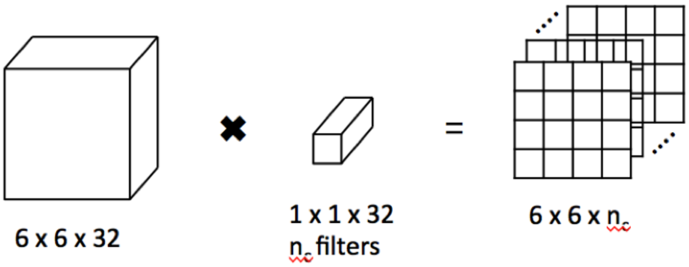
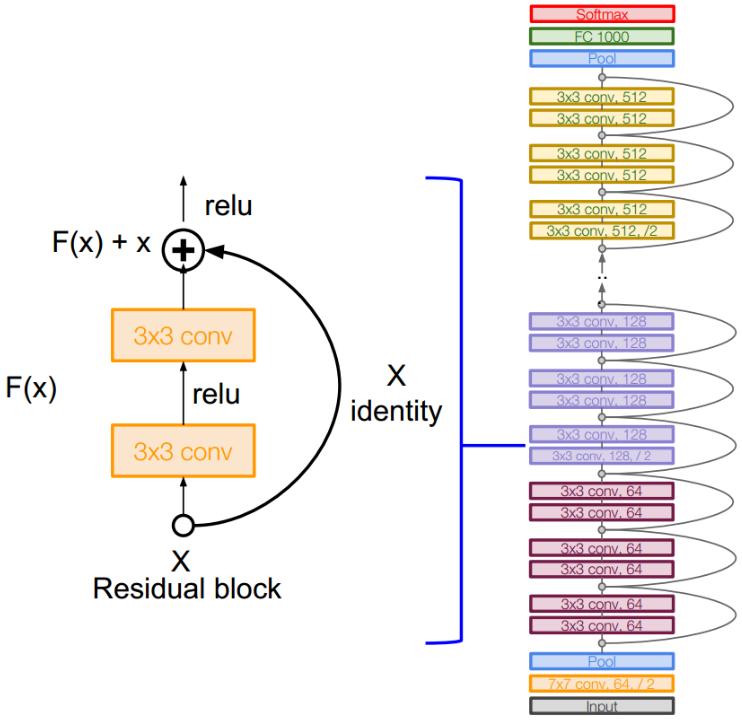


Figure 4.1: An example of an Inception module

- Inception[4] has 22 layers in total
- Deeper networks work better. Networks with too many parameters are prone to overfitting.
- Multiple filter sizes concatenated together.
- “Bottleneck” 1x1 convolutions help prevent depth explosion.
- Fully connected layers are replaced by Average pooling.

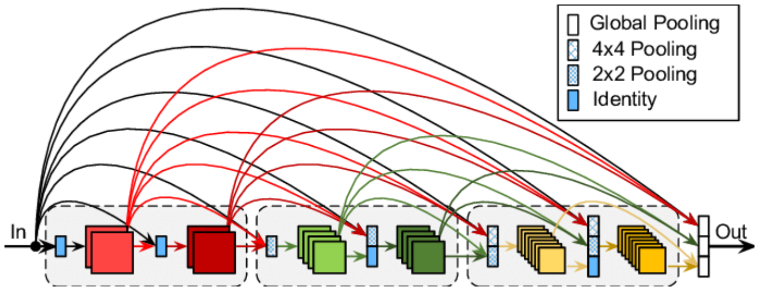


5 ResNet



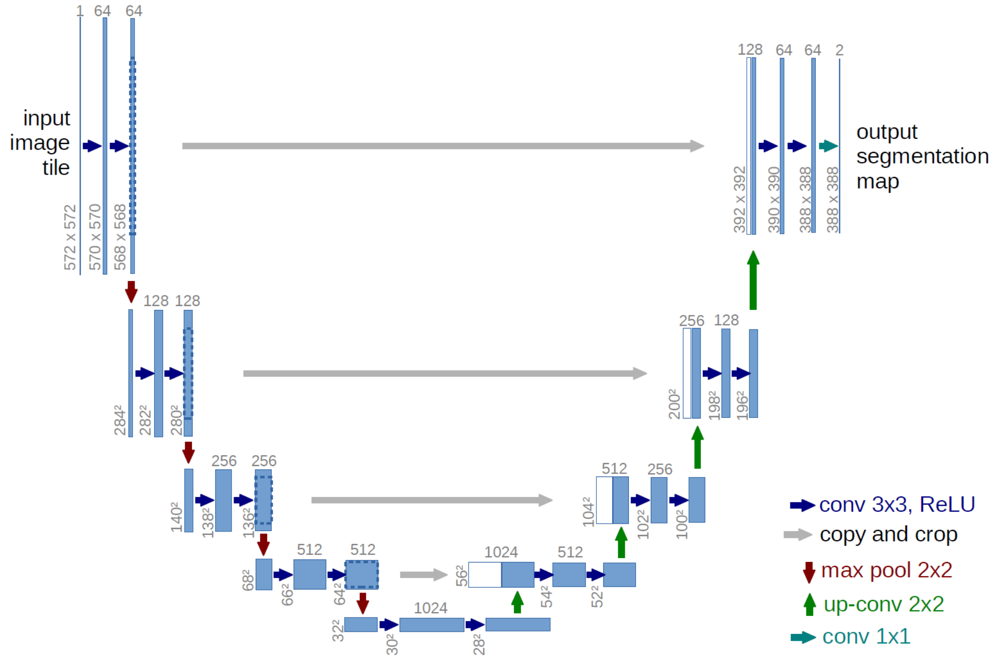
- ResNet[5] is 150 layers deep, 8x VGG!
- Instead of fitting a direct mapping, residual blocks fit the residual mappings.
- Easier to fit identity mappings compared to conventional blocks.
- Use bottlenecks to avoid too many parameters.
- Could be seen as an “information highway” for the gradients. Takes care of the vanishing gradient problem.

6 DenseNet



- In DenseNet[6], each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers.
- Stronger gradient flow.

7 U-NET



- U-Net is a convolutional neural network that was developed for biomedical image segmentation [7].
- Network consists of a contracting path and an expansive path, hence u-shaped architecture.
- During the contraction, the spatial information is reduced while feature information is increased.
- The main idea is to supplement a usual contracting network by successive layers, where pooling operations are replaced by upsampling operators.
- There are a large number of feature channels in the upsampling part, allowing the network to propagate context information to higher resolution layers.

8 TRANSFORMER

- Sequence-to-sequence (seq2seq) models- Used in NLP are used to convert sequences of Type A to sequences of Type B. For example- translation of english sentences to german sentences
- Used in NLP tasks, such as:
  - Machine Translation
  - Text Summarization
  - Speech Recognition

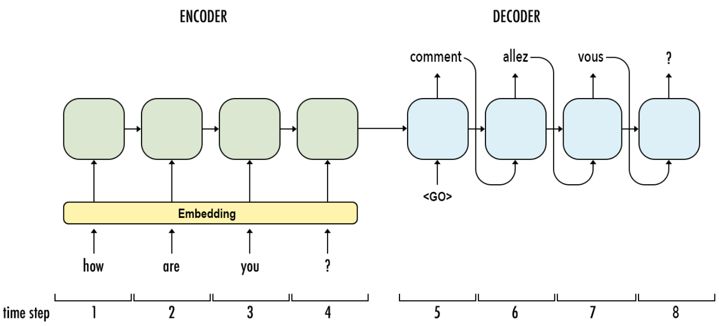
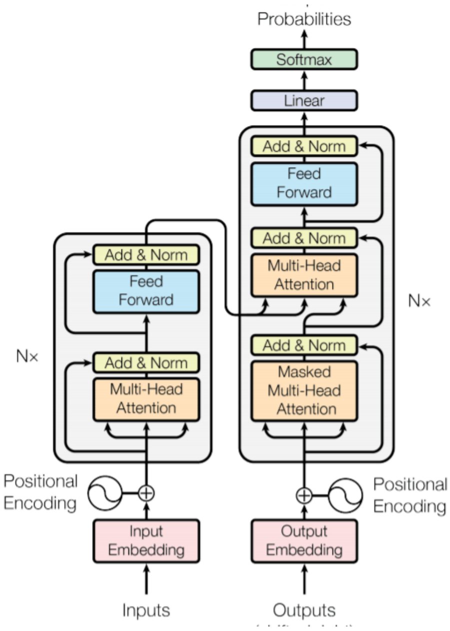
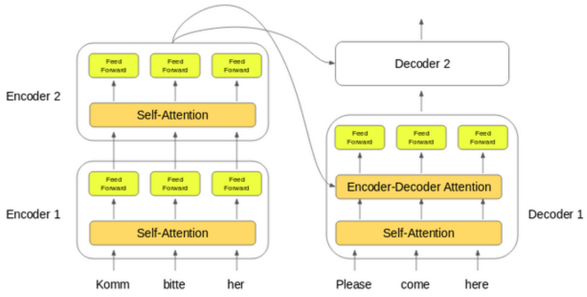


Figure 8.1: RNN based Sequence-to-Sequence Model

- Dealing with long-range dependencies is challenging in RNN.
- Attention model-
  - No longer try encode the full source sentence into a fixed-length vector.
  - Allow the decoder to "attend" to different parts of the source sentence by allowing model learn what to attend based on the input sentence and what it has produced so far.





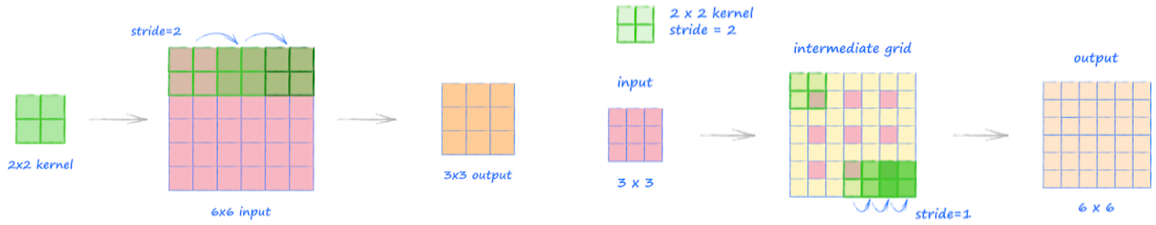
- Transformer [8] aims to solve sequence-to-sequence tasks while handling long-range dependencies.
- Handle the dependencies between input and output with attention and recurrence completely.
- Positional encoding is a re-representation of the values of a word and its position in a sentence.
- Self-attention allows the model to look at the other words in the input sequence to get a better understanding of a certain word in the sequence.
- Shortcoming- can't stretch beyond a certain level due to the use of fixed-length context (input text segments).
- Transformer XL was then proposed to overcome the shortcoming. You can read more about it at `https://arxiv.org/pdf/1901.02860.pdf`.

9 Some tips and tricks

9.1 Picking architectures

- Designing good architectures depends on choosing: kernel sizes, strides, number of channels, pooling sizes, strides, etc.
- Start from a good architecture- LeNet, AlexNet, VGG, Inception, ResNet, WideResNet, DenseNet and then build up
- Any of the above architecture will work for a new problem.

9.2 Upsampling (Deconvolution) operator



(a) Convolution operator

(b) Deconvolution operator

- A typical classification network looks like a pyramid. What if we want to predict larger objects, as in the case of problem of image image segmentation? The input data is a raw image, the annotations/labels in this case are “dense”, e.g., for each pixel in the image you have a label that indicates which class the pixel belongs to (person A, person B, road, car 1, car 2, background etc.) We would like to train a network that predicts a similar segmentation on new images.
- Use a “deconvolution” operator which is also called transposed convolution, convolution with 1/2 strides, “upsampling”, etc. This operator increases the size of the image. You can read more about it at `https://distill.pub/2016/deconv-checkerboard`.

9.3 Weight initialization

- Why initialization matters?
  - If the weights are too small(<1)- Gradient tends to get smaller as we move backward through the hidden layers → neurons in the earlier layers learn much more slowly than neurons in later layers → minor weight updates. This is called vanishing gradient problem.
  - If the weights are too large(>1)- Gradient gets much larger in the earlier layers → extremely high weight updates → overshooting the minimal value. This is called the exploding gradient problem.
- Xavier initialization-
  - Initialize weights by considering characteristics that are unique to the architecture.
  - Constant input/output variance (linear layer): $y = w^\top x \Longleftrightarrow \mathrm{var}(y) = N * \mathrm{var}(w)\mathrm{var}(x)$, assuming i.i.d w_i and x_i and independence between w_i and x_i.
  - Assume inputs are z-normalized, then a reasonable initialization: $w \sim \mathcal{N}\left(0, \frac{1}{N}\right)$.
- Kaiming initialization-
  - Xavier initialization assumes that the activation function is linear, which is clearly not the case.

- Kaiming initialization, which takes activation function into account. For ReLU activation, weights for l^{th} layer are given by:

$$w \sim \mathcal{N}\left(0, \frac{2}{n^l}\right)$$

- Orthogonal initialization for RNN-
  - One of the most extreme issues with recurrent neural networks (RNNs) are vanishing and exploding gradients
  - For illustration, let us look at a simplified RNN model with no inputs, no bias, an identity activation function f and the initial hidden state of the RNN h_0, then

$$\begin{aligned}&h_t = f(Wh_{t-1} + Vx_{t-1}) = f(Wh_{t-1}) = Wh_{t-1}\\&h_n = W^n h_0 = W^n\end{aligned}$$

  Now, let the eigendecomposition of W be $Q\Lambda Q^{-1}$, then $W^n = Q\Lambda^n Q^{-1}$. Following observations can be made:

    - W^n vanishes if the absolute value of all eigenvalues are smaller than 1
    - W^n has a constant norma if the absolute value of all eigenvalues 1
    - W^n explodes if the absolute value of all eigenvalues are larger than 1

  - Due to repeated matrix multiplication within an RNN and the result explodes or vanishes.
  - Use an orthogonal matrix (all eigenvalues 1) to initialize the weights. You can read more about it at `https://smerity.com/articles/2016/orthogonal_init.html`

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.