Chapter 5

Architectures

The field of deep learning has developed over the years for each application domain multiple deep architectures that exhibit good trade-offs with respect to multiple criteria of interest: e.g. ease of training, accuracy of prediction, memory footprint, computational cost, scalability.

5.1 Multi-Layer Perceptrons

The simplest deep architecture is the Multi-Layer Perceptron (MLP), which takes the form of a succession of fully connected layers separated by activation functions. See an example in Figure 5.1. For historical reasons, in such a model, the number of hidden layers refers to the number of linear layers, excluding the last one.

A key theoretical result is the universal approximation theorem [Cybenko, 1989] which states that, if the activation function σ is continuous

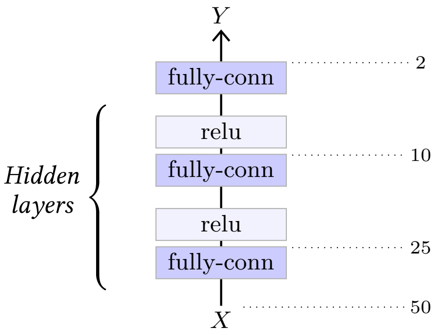


Figure 5.1: *This multi-layer perceptron takes as input a one-dimensional tensor of size* 50*, is composed of three fully connected layers with outputs of dimensions respectively* 25*,* 10*, and* 2*, the two first followed by ReLU layers.*

and not polynomial, any continuous function f can be approximated arbitrarily well uniformly on a compact domain, which is bounded and contains its boundary, by a model of the form $l_2 \circ \sigma \circ l_1$ where l_1 and l_2 are affine. Such a model is a MLP with a single hidden layer, and this result implies that it can approximate anything of practical value. However, this approximation holds if the dimension of the first linear layer's output can be arbitrarily large.

In spite of their simplicity, MLPs remain an important tool when the dimension of the signal to be processed is not too large.

5.2 Convolutional networks

The standard architecture for processing images is a convolutional network, or convnet, that combines multiple convolutional layers, either to reduce the signal size before it can be processed by fully connected layers, or to output a 2D signal also of large size.

LeNet-like

The original LeNet model for image classification [LeCun et al., 1998] combines a series of 2D convolutional layers and max pooling layers that play the role of feature extractor, with a series of fully connected layers which act as a MLP and perform the classification per se (see Figure 5.2).

This architecture was the blueprint for many models that share its structure and are simply larger, such as AlexNet [Krizhevsky et al., 2012] or the VGG family [Simonyan and Zisserman, 2014].

Residual networks

Standard convolutional neural networks that follow the architecture of the LeNet family are not easily extended to deep architectures and suffer

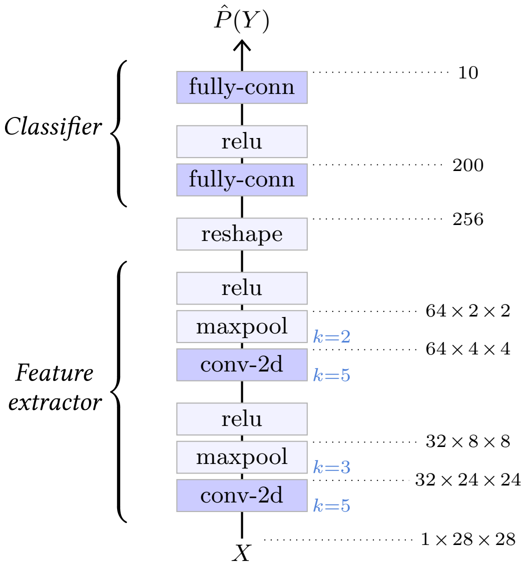


Figure 5.2: *Example of a small LeNet-like network for classifying 28×28 grayscale images of handwritten digits [LeCun et al., 1998]. Its first half is convolutional, and alternates convolutional layers per se and max pooling layers, reducing the signal dimension from 28×28 scalars to 256. Its second half processes this 256-dimensional feature vector through a one hidden layer perceptron to compute 10 logit scores corresponding to the ten possible digits.*

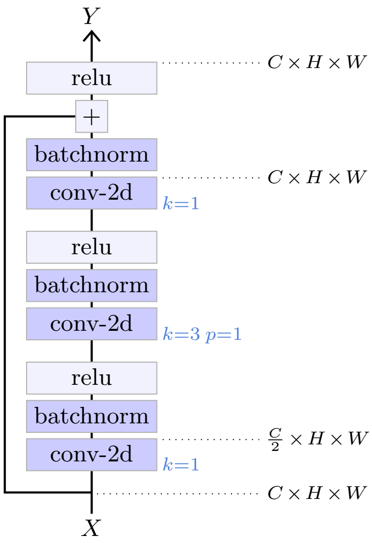
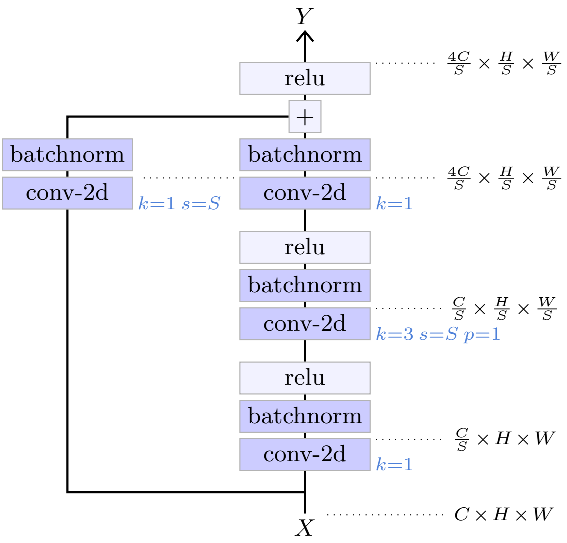


Figure 5.3: *A residual block.*

from the vanishing gradient problem. The residual networks, or ResNets, proposed by He et al. [2015] explicitly address the issue of the vanishing gradient with residual connections (see § 4.7), which allow hundreds of layers. They have become standard architectures for computer vision applications, and exist in multiple versions depending on the number of layers. We are going to look in detail at the architecture of the ResNet-50 for classification.



Y
relu
$\frac{4C}{S} \times \frac{H}{S} \times \frac{W}{S}$
$+$
batchnorm
conv-2d
$k{=}1\ s{=}S$
batchnorm
conv-2d
$k{=}1$
$\frac{4C}{S} \times \frac{H}{S} \times \frac{W}{S}$
relu
batchnorm
conv-2d
$k{=}3\ s{=}S\ p{=}1$
$\frac{C}{S} \times \frac{H}{S} \times \frac{W}{S}$
relu
batchnorm
conv-2d
$k{=}1$
$\frac{C}{S} \times H \times W$
X
$C \times H \times W$

Figure 5.4: *A downscaling residual block. It admits a hyper-parameter S, the stride of the first convolution layer, which modulates the reduction of the tensor size.*

As other ResNets, it is composed of a series of residual blocks, each combining several convolutional layers, batch norm layers, and ReLU layers, wrapped in a residual connection. Such a block is pictured in Figure 5.3.

A key requirement for high performance with real images is to propagate a signal with a large number of channels, to allow for a rich repre-

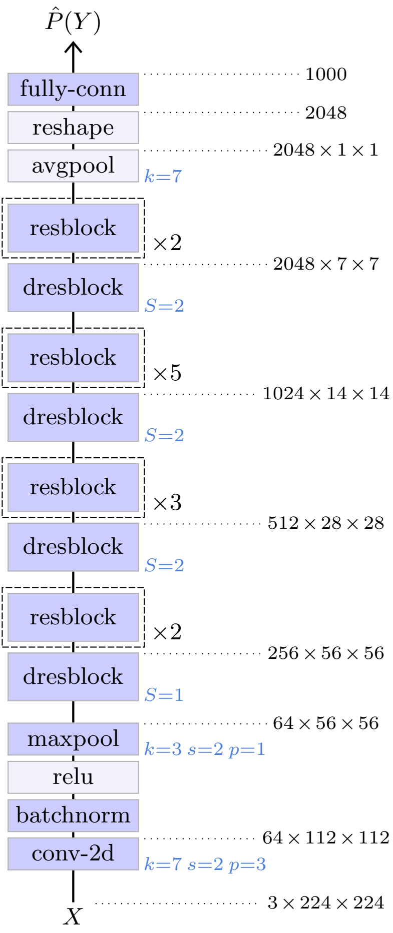


Figure 5.5: *Structure of the ResNet-50 [He et al., 2015].*

sentation. However, the parameter count of a convolutional layer, and its computational cost, are quadratic with the number of channels. This residual block mitigates this problem by first reducing the number of channels with a 1×1 convolution, then operating spatially with a 3×3 convolution on this reduced number of channels, and then upscaling the number of channels, again with a 1×1 convolution.

The network reduces the dimensionality of the signal to finally compute the logits for the classification. This is done thanks to an architecture composed of several sections, each starting with a downscaling residual block that halves the height and width of the signal, and doubles the number of channels, followed by a series of residual blocks. Such a downscaling residual block has a structure similar to a standard residual block, except that it requires a residual connection that changes the tensor shape. This is achieved with a 1×1 convolution with a stride of two (see Figure 5.4).

The overall structure of the ResNet-50 is presented in Figure 5.5. It starts with a 7×7 convolutional layer that converts the three-channel input image to a 64-channel image of half the size, followed by four sections of residual blocks. Surprisingly, in the first section, there is no downscaling, only an increase of the number of channels by a factor of 4. The output of the last residual block is $2048 \times 7 \times 7$, which is converted to a vector of dimension 2048 by an average pooling of kernel size 7×7, and then processed through a fully-connected layer to get the final logits, here for 1000 classes.

5.3 *Attention models*

As stated in § 4.8, many applications, particularly from natural language processing, benefit greatly from models that include attention mechanisms. The architecture of choice for such tasks, which has been instrumental in recent advances in deep learning, is the Transformer proposed by Vaswani et al. [2017].

Transformer

The original Transformer, pictured in Figure 5.7, was designed for sequence-to-sequence translation. It combines an encoder that processes the input sequence to get a refined representation, and an autoregressive decoder that generates each token of the result sequence, given the encoder's representation of the input sequence and the output tokens generated so far.

As the residual convolutional networks of § 5.2, both the encoder and the decoder of the Transformer are sequences of compounded blocks built with residual connections.

- The feed-forward block, pictured at the top of Figure 5.6 is a one hidden layer MLP, preceded by a layer normalization. It can update representations at every position separately.

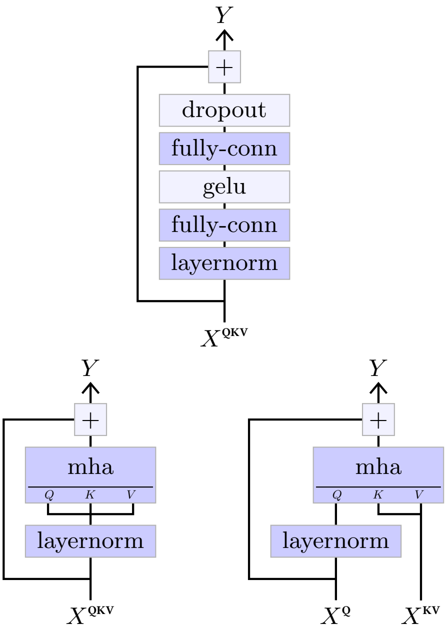


Figure 5.6: *Feed-forward block (top), self-attention block (bottom left) and cross-attention block (bottom right). These specific structures proposed by Radford et al. [2018] differ slightly from the original architecture of Vaswani et al. [2017], in particular by having the layer normalization first in the residual blocks.*

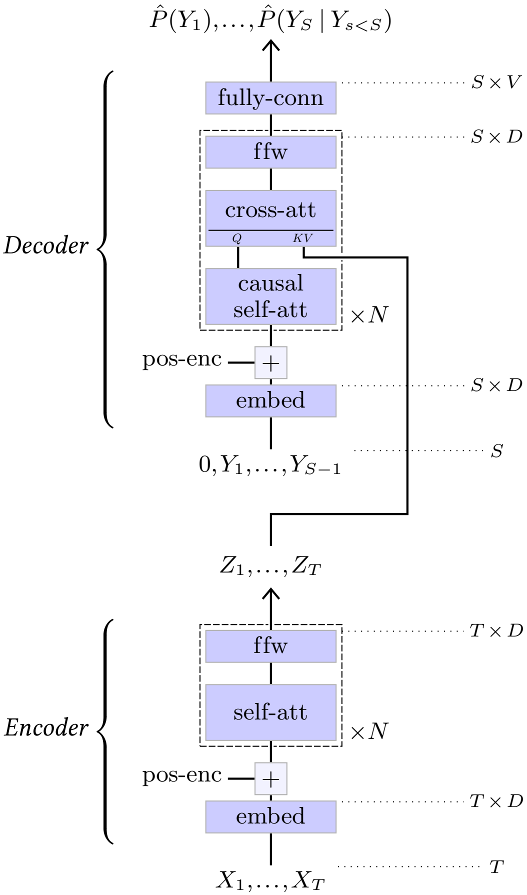


Figure 5.7: *Original encoder-decoder Transformer model for sequence-to-sequence translation [Vaswani et al., 2017].*

- The self-attention block, pictured on the bottom left of Figure 5.6, is a Multi-Head Attention layer (see § 4.8), that recombines information globally, allowing any position to collect information from any other positions, preceded by a layer normalization. This block can be made causal by using an adequate mask in the attention layer, as described in § 4.8

- The cross-attention block, pictured on the bottom right of Figure 5.6, is similar except that it takes as input two sequences, one to compute the queries and one to compute the keys and values.

The encoder of the Transformer (see Figure 5.7, bottom), recodes the input sequence of discrete tokens $X_1,\ldots X_T$ with an embedding layer (see § 4.9), and adds a positional encoding (see § 4.10), before processing it with several self-attention blocks to generate a refined representation $Z_1,\ldots,Z_T$.

The decoder (see Figure 5.7, top), takes as input the sequence $Y_1,\ldots,Y_{S-1}$ of result tokens produced so far, similarly recodes them through an embedding layer, adds a positional encoding, and processes it through alternating causal self-attention blocks and cross-attention blocks to

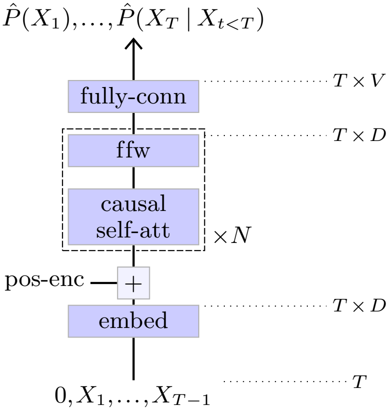


Figure 5.8: *GPT model [Radford et al., 2018].*

produce the logits predicting the next tokens. These cross-attention blocks compute their keys and values from the encoder's result representation $Z_1, \dots, Z_T$, which allows the resulting sequence to be a function of the original sequence $X_1, \dots, X_T$.

As we saw in § 3.2 being causal ensures that such a model can be trained by minimizing the cross-entropy summed across the full sequence.

Generative Pre-trained Transformer

The Generative Pre-trained Transformer (GPT) [Radford et al., 2018, 2019], pictured in Figure 5.8

is a pure autoregressive model that consists of a succession of causal self-attention blocks, hence a causal version of the original Transformer encoder.

This class of models scales extremely well, up to hundreds of billions of trainable parameters [Brown et al., 2020]. We will come back to their use for text generation in § 7.1.

Vision Transformer

Transformers have been put to use for image classification with the Vision Transformer (ViT) model [Dosovitskiy et al., 2020] (see Figure 5.9).

It splits the three-channel input image into M patches of resolution $P \times P$, which are then flattened to create a sequence of vectors $X_1, \dots, X_M$ of shape $M \times 3P^2$. This sequence is multiplied by a trainable matrix W^{E} of shape $3P^2 \times D$ to map it to an $M \times D$ sequence, to which is concatenated one trainable vector E_0. The resulting $(M+1) \times D$ sequence $E_0, \dots, E_M$ is then processed through multiple self-attention blocks. See § 5.3 and Figure 5.6.

The first element Z_0 in the resultant sequence, which corresponds to E_0 and is not associated with any part of the image, is finally processed

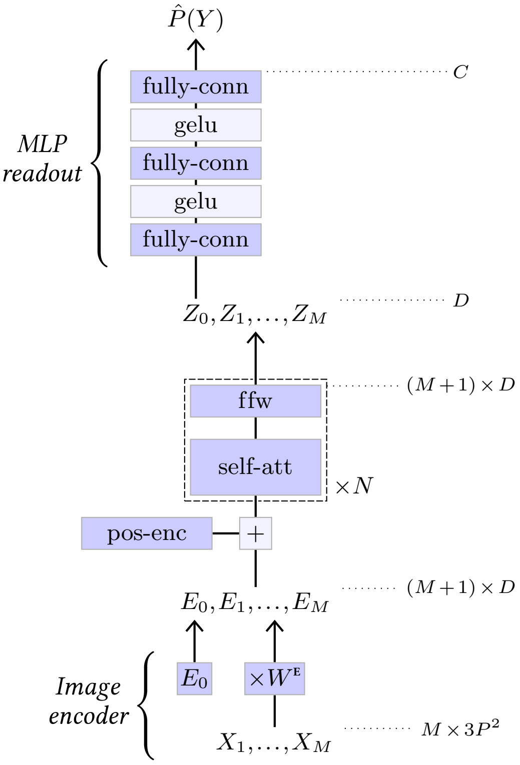


Figure 5.9: *Vision Transformer model [Dosovitskiy et al., 2020].*

by a two-hidden-layer MLP to get the final C logits. Such a token, added for a readout of a class prediction, was introduced by Devlin et al. [2018] in the BERT model and is referred to as a CLS token.

Part III

Applications