Deep Learning - Dry Part

Omer Trinin, Yonatan Azizi

April 2025

1

We expect the accuracy to increase up to a certain h value and then decrease with growth in h. A too small latent space would underfit, i.e., would not capture the sufficiently complex representation required for classification. A too large latent space would overfit, i.e., it would memorize features specific to the training set and would not generalize well to other images.

2

2.a

The UAT theorem states that a neural network, including a network with a single hidden layer, that applies a non-polynomial activation function, can approximate any continuous function with a desired error, given a sufficient number of neurons.

2.b

The conclusion is wrong for the following reasons.

- The theorem states that such a solution exists, but that doesn't mean that a practical algorithm, such as back-propagation, would find that solution.
- The theorem doesn't ensure anything about the ability of the solution to generalize well.
- The theorem doesn't tell anything about the number of neurons required to achieve a desired accuracy, which may be too large if we are restricted to a shallow network.

Deep networks are used since for many problems they can better find a good solution with practical complexity and/or better generalize and/or require a much smaller number of neurons.

3

3.a

The main advantages of CNN for images classification over pure MLP are:

- In images, there are local spatial dependencies between neighboring pixels. While CNN is more restricted than MLP, this restriction makes it easier to extract features based on the local spatial dependencies.
- Especially, due to convolution and pooling layers (or convolution with stride) it is much easier for the CNN to reach invariance to objects' position, rotation and scaling.
- The CNN's usage of many fewer parameters prevents overfitting
- The usage of many fewer parameters also reduces memory and CPU consumption. This is especially important for big images.
- The CNN still allows the use of MLP after extracting important spatial features.

3.b

It is correct that a convolution layer can be viewed as a special case of a fully connected layer. The first difference of convolution layer from fully connected layer is that the convolution inputs are only the neurons in the kernel neighborhood. We can view this feature of the convolution layer as a constraint of having zero weights to all non-local inputs, i.e. inputs outside the kernel neighborhood. The second difference is that the convolution layer applies the same weights across all the input layer. We can view this feature as a constraint on neurons producing an output channel to have the same weights. Overall, a convolution layer is a special case of a fully connected layer. Therefore, theoretically, a fully connected network could emulate a convolution layer. In that sense, CNN can be viewed as adding inductive bias to a fully connected network. Nevertheless, it is better to use CNN, as we need to consider not only when a model can do theoretically, but what can it learns in practice? Since the fully connected network has many more parameters to learn, it is very likely to overfit or stuck in a local minimum without reaching a good solution. Furthermore, the training and the inference would require much more resources. Many empirical experiments indeed shows that it is better to first extract features with convolution layers and only then apply fully connected layers.

4

We expect the algorithm's efficiency to be compromised. By giving higher weights to recent gradients, the EMA algorithm is able to adapt to the changes

in loss function neighborhood as the optimization progresses. In contrast, a linear average would give the same weight to recent and old evaluations, so it would adapt very slowly to changes in loss function neighborhood.

5

Backpropagation computes gradients backward from the loss function to the inputs by applying the chain rule on the computational graph. If the loss was a vector, it should have computed the Jacobian instead of the gradient, but pytorch is not designed to compute the Jacobian. Furthermore, there is a more fundamental problem of how to do the optimization step given a computation of multiple directions, one for each component of the loss vector? This is not a well-defined problem, unless we are provided with the way the different components of the vector should be combined. But then, if we are given how the loss vector components should be combined, we would better combine them accordingly into a scalar loss and proceed as usual with standard backpropagation.

6

6.a

A learning algorithm is aimed at providing correct outputs for inputs not shown during training. Since there are many possible ways to generalize the training data, this is not a well-defined problem, unless we add additional assumptions about the target function. An example of such an assumption is assuming the target function to be smooth. Such assumptions are usually inherent in the algorithm design. These assumptions are called inductive bias, i.e. the set of assumptions that determines the preferences of the algorithm for how to make predictions on new data based on the training data.

6.b

One bias of CNN is spatial locality as each kernel detects only local patterns between neighboring pixels. A second bias is translation equivariance as the same filter with the same weights is used across the entire layer.

6.c

Inductive bias restricts the learning algorithm and the model based on prior knowledge about the target function. The pros of that are avoiding overfitting, reducing computational costs, and being able to reach a good model with relatively little training dataset. The cons are that the bias requires prior knowledge we don't always have and that these assumptions are often approximations that limit the model quality. For these reasons, we would prefer avoiding inductive bias when we don't have sufficient prior knowledge about the target and when

we have sufficient data and computational resources to learn a good model with less inductive bias. Conversely, we would better use inductive bias when we have prior knowledge about the target function, while we don't have sufficient data or sufficient computational resources to learn a good model without using the prior knowledge.

7

7.a

The sequence length is n.
The dimension of the vector representations is d.
The dimension of Q and K matrices is $n \times d$. Therefore, the time complexity of the multiplication QK^T is $O(n^2d)$
The multiplication result is an $n \times n$ matrix. Therefore, the time complexity of dividing it by the scalar $\sqrt{d}$ is $O(n^2)$.
The softmax costs n operations on each row of an $n \times n$ matrix. Therefore, its time complexity is $O(n^2)$.
The time complexity of the multiplication of the resulting $n \times n$ matrix by the $n \times d$ matrix V is $O(n^2d)$.
The sum of all these is $O(n^2d) + O(n^2) + O(n^2) + O(n^2d) = O(n^2d)$
Therefore, overall, the computational time complexity of computing the output of the attention layer is $O(n^2d)$

7.b

The time complexity of dividing Q by the scalar $\sqrt{d}$ is $O(nd)$
The softmax costs d operations on each row of the resulting $n \times d$ matrix. Therefore, the time complexity of dividing by $\sqrt{d}$ is $O(nd)$.
The time complexity of the multiplication K^TV is $O(nd^2)$ since it is a multiplication between $d \times n$ and $n \times d$ matrices.multiplication
The last multiplication is between $n \times d$ and $d \times d$ matrices. Therefore, its time complexity is $O(nd^2)$
The sum of all these is $O(nd) + O(nd) + O(nd^2) + O(nd^2) = O(nd^2)$
Therefore, overall, the computational time complexity of computing this method is $O(nd^2)$
Given $d << n$, this indeed helps reducing the computational complexity from $O(n^2d)$ to $O(nd^2)$
However, this computation doesn't preserve the function of the original attention formulation, since Softmax is not a linear operation.
This computation is likely to lead to bad results, since it ruins the computation of the attention scores, by applying Softmax on the query, before multiplying it by the key.

8

8.a

Each pixel represents the cross-attention weight from the English word of the column to the French word of the row. The brighter the pixel, the higher the weight, starting from black to zero up to white for 1.

8.b

A row with a single non-zero pixel means that the French word of that row requires attention to only a single English word of that column. In the example, this happens with words related to a year, a month, and a place. Indeed, these words could be translated from a single word into a single word, with no need for the context of surrounding words.

8.c

A row with several non-zero pixels means that the translation into that French word was translated with attention to several words in the source English sentence. That may happen for several reasons, such as translation dependency on the context of some surrounding words, a multi-word expression translated into a single word, or words reordering. For example, in the translation of "the European Economic Area", the attention of the French word "la" is to the source English word "the", but also to the word "Area" to which the word "the" is related to (and in French this word should appear right after "the").

8.d

The reason is that due to the Softmax, the sum of weights in a row is 1. Therefore, a pixel in a row is white ($weight = 1$) iff the other weights in the row are black ($weight = 0$). On the other hand, if some pixel in a row is gray ($0 < weight < 1$) then there is at least one more gray pixel ($0 < weight < 1$) in the row to complete the weights sum into 1.

9

9.a

The mathematical basis we rely on when we ignore the KL-divergence term is that the KL-divergence is non-negative and therefore the ELBO term $E_q[log(\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)})]$ is a lower bound of the marginal log-likelihood $log(p_\theta(x_0))$. The reason we maximize this lower bound, rather than the marginal log-likelihood, is that we can't compute the KL-divergence term.

9.b

To compute the KL-divergence

$$D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)) = \sum_{x_{0:T}\in\mathcal{X}} q(x_{1:T}|x_0) log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)})$$

We need to compute $p_\theta(x_{1:T}|x_0)$, which is hard to compute since we don't have a closed formula for it, but rather we would need to estimate it numerically based on samples, while running an NN model for computing each sample.

9.c

The prior $p_\theta(x_T)$ is set to a standard normal distribution. Unlike VAEs, in diffusion models, the forward process is a non-learnable process, during which noise from a fixed distribution is gradually added. Therefore, this process results in a fixed distribution. Therefore, $D_{KL}(q(x_T|x_0)||p_\theta(x_T))$ is a distance between two fixed distributions. Therefore, it is **constant**, so it can be ignored.

10

10.a

One problem with Bob's is that the autoencoder provides no guarantee about the distribution of the latent space. Therefore, a random vector may be way off from any digit. Furthermore, even if the random vector is within the space enclosing the vectors corresponding to the trained data, the results may be bad, since there is no guarantee of a smooth transition between the vectors corresponding to the training data. Specifically, the result may not be similar to any one of the 10 digits.

10.b

The VAE solves the problems described above by two modifications. First, instead of mapping input data into a single point in the latent space, the encoder maps the input into a normal distribution (or more accurately, into parameters of a normal distribution, which enable us to compute the gradient with respect to these parameters during training, while still sampling with the reparameterization trick). Second, it adds to the reconstruction error loss function a KL-divergence term. The KL-divergence term regularizes the latent space to follow a standard normal distribution. These changes increase the reconstruction error of the trained data but enable the model to better generalize and achieve smooth transitions between different digits.