**Dry Assignment**

**Mayan Rivlin Gilboa 322515438, Dor Knafo 315810317**

1. We can categorize the range into three segments: one where the latent dimensions are very small, another where they are very high, and the last where they are optimal (neither too low nor too high). In the first segment, where the latent dimensions are very small, we expect the model to perform poorly because it lacks the capacity to learn complex data structures. A dimension that is too low limits the model's ability to capture intricate patterns, likely resulting in low classification accuracy. When the latent dimension is too high, beyond a certain threshold, the model may gain excessive capacity to memorize the training data. This can hinder its ability to generalize to new, unseen data, causing classification accuracy to decrease as the dimension grows too large. In the "optimal" range, between the extremes of too small and too large dimensions, the model is expected to perform best on unseen data. This range strikes a balance between the model's ability to learn complex patterns without overfitting to the training data, leading to better generalization and improved classification accuracy overall.
2.
    a. The Universal Approximation Theorem (UAT) states that a neural network with a single hidden layer can learn to represent any continuous function on a compact domain to an arbitrary degree of accuracy, given suitable activation functions. This means that neural networks can approximate any real-world continuous function, no matter how complex, by adjusting the weights, number of neurons, and biases within the network, although in practice this might require a very large number of neurons..
    In other words, theoretically, you don't need deeper networks (more hidden layers) since according to the UAT a single-layer network can always approximate the optimal solution.
    Thus, the UAT supports our friend's claim that using an MLP with a single hidden layer can, in principle, achieve optimal error over any dataset and loss criterion.
    b. While the Universal Approximation Theorem guarantees that a single-hidden-layer MLP can, in theory, approximate any continuous function given enough neurons, it does not address the following key practical factors:
        - Single-layer MLPs can (theoretically) approximate any function, but they might require an extremely large number of neurons to achieve the same accuracy as a deeper MLP, making them significantly less efficient. This is because networks with multiple hidden layers can learn and extract hierarchical features more compactly, resulting in fewer parameters and a more manageable model size.
        - It is also likely harder to train a single layer with a very large number of neurons than it is to train a deeper network with several, smaller layers.

Modeling complex functions using just one layer is inherently difficult and can demand a prolonged training process. In contrast, deeper architectures can distribute the complexity across multiple layers, making use of more effective gradient flow and enabling each layer to tackle a simpler sub-problem, thus often resulting in more efficient and faster training.

- Another advantage of going deeper is better generalization. By learning a structured hierarchy of features, deeper models are more adaptable to new data and therefore usually perform better on a range of tasks.

To conclude, while a single-layer MLP can, in theory (according to the UAT), approximate any function, in practice deeper networks are usually more efficient, easier to train, and better at generalizing, making them more suitable for complex tasks.

3.

a. Bob can convince Alice by explaining her the following points:
First, an advantage CNN has over MLP in the context of image classification relates to spatial structure and locality. Images naturally have spatial structure, meaning that nearby pixels tend to be more related to each other than distant ones. CNNs exploit this structure by using convolutional layers, which apply filters that slide over local regions of the image, thus detecting edges, textures, and other local features. In contrast, an MLP treats each pixel as an independent input, ignoring spatial relationships. This makes MLPs less efficient for image classification because they struggle to capture important local patterns like edges, corners, or textures.
Another key advantage CNNs have over MLPs that can help Bob convince Alice is the number of parameters. A CNN's parameter count is determined by the filters themselves—only their fixed size and quantity matter—since each filter is applied repeatedly across the image, regardless of the image size. By comparison, each neuron in an MLP must connect to every input pixel (fully connected), leading to a massive number of parameters, especially for large images.

Challenges Alice might face include a high computational cost and poor generalization...
MLP will require a large number of parameters she would have to train and increase the risk of overfitting—particularly when data is limited.
Another challenge stems from the first advantage mentioned: loss of spatial information.
Moreover, MLPs flatten images into a one-dimensional vector, destroying the important spatial structure in the data. Without the ability to recognize spatial features, MLPs struggle to generalize to new data, especially if features appear in varied locations or scales.

b. We don't agree with Alice. It's not correct to treat any linear operation as if it were the same. While convolutions are indeed linear transformations (applied before a nonlinear activation), they differ from typical linear layers in MLPs in two crucial ways:
   - Local Features and Spatial Structure:
     CNNs focus on local features (like edges and textures) and naturally capture the spatial structure of images. In contrast, MLPs flatten images, treating each pixel as an independent input, which typically requires more parameters and more time to learn local relationships.
   - Weight Sharing:
     A single filter in a CNN is applied to different regions of the image to detect the same feature (e.g., edges) in various locations. This weight sharing reduces the number of parameters compared to MLPs, where each neuron has a unique set of weights.

   To conclude, we disagree with Alice because convolutions are more than just a linear operation in the context of image classification. After the convolution, non-linear activations are applied, enabling complex non-linear mappings. In the bottom line, it's not just about linearity but about how the architecture leverages that linearity. CNNs exploit image properties like locality and spatial structure in ways that MLPs simply cannot.

4. Yes, we would expect the model's efficiency to be compromised.
   EMA primarily differs from linear averaging of accumulated gradients in how it assigns weight to newer versus older gradients. The key advantage of EMA is its emphasis on more recent gradients, while older gradients gradually lose their influence. This decay allows EMA to adapt more quickly to changes in the optimization direction, making it less sensitive to noise (as noise is reduced over time) and facilitating faster convergence.
   In contrast, linear averaging assigns equal weight to all gradients throughout the optimization process. As a result, older, less relevant gradients continue to influence the model, making it harder for the model to adjust to changes in the optimization direction. Moreover, linear averaging is more sensitive to noise, as it treats noise the same as any other gradient, which can slow convergence and introduce unnecessary fluctuations.
   Overall, replacing EMA with linear accumulation would likely hinder the optimization process, leading to longer training times and potentially preventing the model from reaching the optimal minimum.

5. In PyTorch, the loss tensor needs to be a scalar for loss.backward() to function properly and compute gradients. Backpropagation depends on the gradient of a single scalar to apply the chain rule across all layers in the network. When the loss is a scalar, the gradient can be efficiently calculated by traversing the computational

graph. However, if the loss is not scalar, it leads to confusion when determining how to compute gradients across the various dimensions. Loss functions such as mean squared error or cross-entropy are specifically designed to output a single scalar value to prevent this issue. Without a scalar loss, PyTorch wouldn't be able to correctly perform gradient propagation.

6.

a. Inductive bias can be defined as the set of assumptions or biases that a learning algorithm employs to make predictions on unseen data based on its training data. These assumptions are inherent in the algorithm's design and architecture and serve as a foundation for learning and generalization.
The inductive bias of an algorithm influences how it selects a hypothesis from the hypothesis space that best fits the training data. It helps the algorithm navigate the trade-off between overfitting and underfitting.
In other words, inductive bias is the "built-in" or "preferred" way a model searches for patterns, which influences how it extrapolates beyond the training examples.

b. CNNs encode several inductive biases that make them particularly effective for image-related tasks. Two notable examples of these biases are:
   - Local Receptive Fields: CNNs use small, local receptive fields to capture patterns in different parts of the input. This is based on the assumption that meaningful patterns can be represented by combinations of local features.
   - Weight Sharing: CNNs use weight sharing through convolutional kernels, which means that Instead of learning separate weights for every possible location in the image, CNNs reuse the same set of filter weights across all spatial positions. This encodes the belief that the same features (edges, textures) are relevant throughout the image.

c. When the bias aligns with the underlying data structure—such as the local feature assumptions in CNNs for image tasks—it typically enhances generalization, reduces the amount of data required, and speeds up training. However, if these assumptions do not match the actual data patterns, they can constrain the model too much and delay its ability to capture important relationships. Stronger inductive biases are especially beneficial in domains with limited data or well-understood properties (e.g., images, sequences), whereas less biased and more flexible models are often preferred when you have large datasets or a domain with unknown or highly varied structures.

7.

a. Forming the n*n attention matrix : Q·K^T (An n×d * d×n) ->
   costs O(n·d· n) = **O(n^2 · d).**
   Next, we apply the softmax operation on the n×n matrix, which is O(n^2).
   Multiplying the attention matrix (n×n) by V (n×d matrix) also costs O(n^2 · d).
   The overall time complexity is **O(n^2·d).**

b. By computing K^T·V first, Bob moves one of the “expensive” O(n^2· d)matrix multiplications down to a smaller O(n d^2) operation (because K∈Rn×d and V∈Rn×d yield a d×d product). This indeed reduces complexity (assuming d≪nd). However, it does not preserve the original attention mechanism, because it no longer computes pairwise interactions (token to token weighting) as in the classic Softmax(Q·K^T)·V formulation.

Therefore, while Bob’s approach lowers time complexity it is not equivalent to the original attention formulation and loses the core idea which makes self attention effective.

8.

a. Each pixel in the attention map represents the attention weight assigned by the model when translating a specific word from English to French. The x-axis corresponds to the source language (English), and the y-axis corresponds to the words in the generated target sentence (French). We can see that some pixels are brighter than others, which means the model is paying more attention to that source word when generating the corresponding target word.
b. Having rows with only one non-zero pixel in the attention map means that each target words in that word attends to only a single source word in high confidence. This indicates a one-to-one alignment between the source and target words.
c. Having rows with more than one non-zero pixel in the attention map indicates that each target word attends to multiple source words simultaneously. This usually happens when translating **a** sentence that requires context from more than one word in the source language to form a coherent target word or phrase.
d. As we mentioned before, having rows with a single non-zero pixel indicates that each target word in that row attends to only a single source word with high confidence, so the related pixel gets all the weight and will be colored white. In rows where we have more than one non-zero pixel, the weights will be distributed among the words (pixels), and we will no longer see a white pixel but rather a few gray pixels that share the total weight. This is because having multiple non-zero pixels indicates that each target word attends to multiple source words simultaneously.

9.

a. The mathematical basis for ignoring the KL-divergence term is the assumption that the posterior distribution is sufficiently close to the prior distribution. Since KL-divergence measures the difference between these distributions, it is assumed to be small enough that it does not significantly contribute to the loss function, allowing us to focus on optimizing the ELBO term.
b. The primary reason we cannot compute the KL-divergence term is that it involves an intractable integral. The posterior distribution p(z|x)p(z | x)p(z|x) is

typically computed using Bayes' rule, where the denominator is an integral over all possible latent variables, which is often impossible to evaluate. When the latent variable is high-dimensional, calculating the KL-divergence requires integrating over a complex probability space, making it computationally infeasible.

c. In general, we do not ignore this term during training, as it plays a crucial role in regularizing the latent space. However, in some specific cases, it might be negligible or not actively optimized. If the prior p(z) is a standard normal distribution, and the variational posterior p(z|x) is a normal distribution the KL divergence term has a well-known closed-form solution. In such case we can compute it directly and treat it as a constant in the loss function. During training, we aim to maximize the ELBO, which means minimizing the KL term while maximizing the reconstruction likelihood. However, since the KL term is analytically solved and non-trainable, it does not require explicit gradient-based optimization.

10.

a. Bob is incorrect because a self-supervised autoencoder is not a true generative model like VAEs or GANs. In the autoencoder we trained for our wet assignment, the encoder compresses an image into a 128-dimensional latent vector, which serves as a compact representation of the MNIST image. The decoder then reconstructs the original image from this vector. However, this autoencoder is not designed to enforce a well-defined probability distribution (such as a Gaussian) in the latent space. As a result, randomly sampling vectors from an arbitrary distribution does not necessarily generate meaningful MNIST-like images when fed into the decoder.

b. A Variational Autoencoder (VAE) is a type of autoencoder designed for generative modeling. Unlike a standard autoencoder, which focuses solely on compressing and reconstructing data, a VAE learns a probabilistic latent space, enabling it to generate new, meaningful samples by drawing from a known distribution.

   The key difference between a VAE and our standard autoencoder is how the encoder processes inputs. In our standard autoencoder, the encoder maps an input image to a fixed latent representation, which the decoder then uses to reconstruct the original image. In contrast, a VAE's encoder outputs a distribution (characterized by a mean μ\muμ and variance σ2\sigma^2σ2) over possible latent representations. This ensures that similar inputs map to nearby points in the latent space, making it more structured and continuous.

   This modification allows VAEs to function as true generative models. Since the latent space follows a well-defined distribution, we can generate new images by sampling random points from this distribution and passing them through the decoder. Unlike standard autoencoders, VAEs ensure that every point in the

latent space corresponds to a meaningful output, making them well-suited for generative tasks.