# Theoretical Questions:

Sajjad Zangiabadi

## Question 1:

- AutoEncoders fall into the category of neural networks designed for specific jobs such as learning key features and making data less complex in terms of dimensions. These networks have an encoder part which squeezes the input data into a hidden layer and a decoder part that brings back the original input from that hidden layer. The kind of AutoEncoder gets defined by how its hidden layer size stacks up against the size of its input layer.

- Under Complete AutoEncoders:
  - Latent Space Dimension < input dimension.
  - Advantages include its ability to squeeze data into smaller spaces effectively. It shines in trimming down complex data sets as seen in techniques such as Principal Component Analysis and excels when dealing with data that contains repetitive information.
  - When it comes to storing and sending data quickly especially on smartphones image compression works wonders.

- Over Complete AutoEncoders:
  - Latent Space Dimension > input dimension
  - Can grasp more complicated and non-linear connections found within the data. They prove to be invaluable tools for cleaning up data by getting rid of the noise and for spotting the odd ones out.
  - In the realm of network security, one finds that spotting anomalies stands out as a method to pinpoint what doesn't align with usual patterns.

- Exactly Complete AutoEncoder:
  - Latent Space Dimension = input dimension.
  - Advantages include being easy to train and acting as a standard for comparing with both less complex and more complex designs.
  - In most cases scenarios don't find their way into everyday use, instead they serve as markers to grasp how shifts in dimensionality play out.

## Question 2:

- AutoEncoders and models based on traditional supervised learning follow different paths especially when it comes to how they use data with labels. The goal of AutoEncoders is to get good at showing data in a simpler way. This involves using an encoder to make the input smaller and then a decoder to put it back together. The goal of their training is to keep the reconstruction error as low as possible. They usually use ways to measure this like the Mean Squared Error. Since this method doesn't need data with labels the AutoEncoders work well for finding things that don't fit in data compression and pulling out important features.
- Looking at it from another angle, the goal of conventional models trained under supervision is to connect the dots between what's given and the expected outcomes, trying to guess or sort them out correctly. On the flip side, these models lean heavily on data that's already been tagged with the right answers. They use tools like the Mean Squared Error when they're dealing with numbers that need to fall into a specific range and Cross-Entropy Loss when they have to put things into their correct categories, all in an effort to get as close as possible to the mark. How well supervised models do depends a lot on having enough good quality labeled data.
- In situations where there's not much labeled data around supervised learning models get creative. They mix a little bit of labeled data with a lot of unlabeled data in what's known as semi-supervised learning. Then there's transfer learning where they take models already trained on similar stuff and use them. Plus there's active learning which is all about picking out the data samples that tell us the most and labeling those. By taking these steps they tackle the problems that come with not having enough labeled data head-on making it easier for the model to learn better and faster.

## Question 3:

- The reconstruction loss function is essential for assessing the model's ability to reconstruct input data from its compressed representation during AutoEncoder training. Mean Squared Error (MSE), Binary Cross-Entropy (BCE), and Kullback-Leibler Divergence (KL Divergence) are the three often utilized loss functions.

- The Mean Squared Error or MSE works by figuring out the average of the squared differences between what was originally there and what has been reconstructed. Because it's easy to put into practice and it ensures the gradients stay steady it's a go-to option when dealing with data that keeps on changing. Despite its advantages mean squared error does not handle outliers well. These outliers can throw off the calculations leading to less-than-ideal results when the data is noisy.

- The difference between the true and predicted binary distributions is measured using the Binary Cross-Entropy (BCE) technique. For binary or categorical data, where each input is represented as a probability between 0 and 1, this loss function is perfect. Compared to MSE, BCE is less sensitive to outliers; yet, it can become unstable when probabilities get very near to 0 or 1, producing huge gradients that might cause problems with optimization.

- In the realm of Variational AutoEncoders or VAEs for short the concept known as Kullback-Leibler Divergence or simply KL Divergence steps into the spotlight. Its role? To measure how much one probability distribution diverges from another. This mechanism acts as a lens offering a view into how well the distribution being learned mirrors the actual one.

- In the process of selecting a reconstruction loss function one must pay attention to the kind of data involved and what the task demands specifically. For data that flows continuously MSE stands out as a simple yet effective choice. Binary Cross-Entropy works well when dealing with data that's either this or that but it's important to be cautious with very high or low chances. Despite needing more computing power KL Divergence shines when you're trying to match up or make sense of different probabilities.

- $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{x_i})^2$
- $\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[x_i \log(\widehat{x_i}) + (1 - x_i)\log(1 - \widehat{x_i})]$
- $\text{KL}(P \parallel Q) = \sum_{i=1}^{N} P(x_i) \log \frac{P(x_i)}{Q(x_i)}$

## Question 4:

- When one looks into how good the learned info is in an AutoEncoder without having any tagged data they dive into a few key areas: how well it can rebuild what it sees the features of the hidden layer how it groups similar things and how tough it is against errors.

- To understand how well an AutoEncoder is working one common approach involves looking at the mistakes it makes when rebuilding data. This is often done by checking the average of the squares of the errors. When the numbers showing the errors in rebuilding are small it usually means the AutoEncoder is doing a good job at noticing the key parts of the data. Looking at the original and rebuilt samples with your own eyes particularly when dealing with pictures can help you understand how well they were reconstructed.

- When people use methods such as t-SNE or UMAP to create pictures of the latent space it shows how good the AutoEncoder is at understanding the structure of the data. If you see groups of data points clearly apart from each other in these pictures it means the representations are good.

- When one investigates the world of clustering analysis it becomes clear that diving into the latent space with tools like K-means or DBSCAN sheds more light on the topic. As for understanding how well these clusters hold together the silhouette score steps in as a handy metric. A bigger number here means the clusters stand apart more clearly. Seeing these groups laid out makes it easier to grasp how the hidden space is arranged.

- When one takes a close look at the spread of hidden variables by plotting them on histograms it becomes possible to spot any patterns that might raise eyebrows. By applying methods like what's found in Variational AutoEncoders or VAEs for short it's possible to maintain a latent space that's both smooth and full of significance. Seeing how this kind of control impacts the quality of what's being represented can offer some deep insights.

- By applying the learned insights to different tasks like transfer learning or spotting oddities one can indirectly figure out how good they are. Also by seeing how these insights hold up when the data gets messy or weird we can tell how tough they are. Exploring the space right next to certain points and seeing how things blend together using qualitative methods can really help someone understand if what's being learned makes sense and stays the same across different situations.

## Question 5:

- In the world of Variational AutoEncoders or VAEs for short the reparameterization trick is a game changer. It lets gradients flow backward through the sampling step a must-have for the learning phase. Instead of pulling samples straight from the distribution labeled N with μ(x) and $\sigma^2(x)$ as parameters this clever technique opts for a simpler path. It picks from a plain old normal distribution marked as ϵ∼N(0, I) and then shifts and scales it with z equals μ(x) plus σ(x) times ϵ. The way this change can be broken down allows for the smooth passage of gradients. $(z=\mu(x)+\sigma(x)\cdot\epsilon)$

## Question 6:

- A hyperparameter called β is added to the standard VAE in the β-VAE variation in order to balance the trade-off between the latent space's complexity and reconstruction accuracy. The Kullback-Leibler (KL) divergence factor is weighted more strongly when β increases, which pushes the latent space closer to the prior distribution, which is typically a standard normal distribution.
- $L = E_{q(z|x)}[\log p\,(x|z)] - \beta \cdot D_{KL}\big(q(z|x)|p(z)\big)$

## Question 7:

- When one tries to train Variational AutoEncoders or VAEs on datasets that are complex and high-dimensional they often hit problems such as mode collapse issues with the posterior not holding up and struggles in grasping the core structure of the data. To address these problems a variety of tactics can be put into action. By diving into deeper networks or turning to convolutional structures one can boost the model's ability to grasp intricate patterns. On another note, tweaking the objective function of the model with tactics like adversarial training or adding extra losses helps avoid repetition and adds to the variety of outcomes. By applying methods such as disentangled representation learning alongside importance sampling and tweaking the annealing schedules for the KL divergence term it becomes possible to make the training process more stable. This helps in stopping the posterior collapse issue allowing for a more efficient training of VAEs especially when working with complex datasets.

## Question 8:

- It's possible to add more rules and tweaks to Variational AutoEncoders making them even more useful for different jobs. By introducing conditional generation to VAEs they gain the ability to create examples based on certain features or tags. It lets people manage the data they create more carefully. This is great for when they need to make images that have specific features. When you let semi-supervised learning work with VAEs it takes in both the data that's been tagged and the data that hasn't. This way it gets better at applying what it knows across different situations because it learns a stronger way of recognizing patterns. By weaving together adversarial training methods with Variational Autoencoders or VAEs the approach known as Adversarial Variational Bayes or AVB steps up the game. It brings to the table samples that boast clearer details and stick closer to the real data's distribution. This boost in quality makes them a go-to choice for jobs that need the creation of lifelike samples like crafting images from scratch. Bringing in these upgrades opens up new doors for Variational Autoencoders making them more flexible and powerful tools for a variety of real-world uses.

## Question 9:

- Standard Convolutional Layer: A kernel traverses the input in this basic convolutional layer to create feature maps. It is extensively employed in applications such as object detection and image categorization.

- Dilated Convolutional Layer: By creating gaps in the kernel, dilated convolutions can collect a greater range of contextual information without requiring an increase in parameters. They are helpful in applications such as picture segmentation, where they maintain spatial resolution and capture the overall context.

- Transposed Convolutional Layer (Deconvolution): This layer upsamples input feature maps and is frequently employed in GANs to produce high-resolution images or in designs such as U-Net for image segmentation.

- Depthwise Separable Convolutional Layer: This reduces processing and model size by splitting convolution into depthwise and pointwise stages. Ideal for situations with limited resources, such as those involving mobile devices.

## Question 10:

- In the world of convolutional neural networks, there's a clever trick known as dilated convolution. This method spaces out the elements of the convolutional kernel. By doing so, it can take in a wider area without the need to cram in more parameters.

- It finds its application in:
  - Expanding the area of focus allows for a wider understanding without making things more complicated.
  - Cutting down on the number of parameters: It's about making the area that receives input bigger but without adding a lot more parameters.
  - In the process of extracting features on multiple scales one finds that it involves capturing these features at various levels all at once.

- In situations found in real life

  - In the realm of self-driving cars the practice of semantic segmentation plays a key role in recognizing objects scattered across the streets.
  - In the world of medical image analysis the focus often lies on spotting tumors or dividing images into sections that show different organs.
  - In the field of Natural Language Processing they are working on getting better at figuring out the context when it comes to jobs such as figuring out feelings in text.