Q1-

In the context of AutoEncoders, the terms "under complete", "overcomplete", and "exactly complete" refer to the capability of the bottleneck layer (also called the latent space or embedding layer)for input data. Here's a quick explanation of each:
Undercomplete autoencoders: In this variant of AE's, the dimension of latent space is less than the input data, it leads the model to learn compressed representation of the input data. It can help with dimensionality reduction and also feature extraction.
The UAE's are useful for these scenarios:

- Dimension reduction.
- Learning only compact representations of data instead of copying them!
- Feature extraction.

On the other hand, it could lead to loss of information and cause low-quality generation.
Overcomplete AE's: unlike before, in this variant of the AE's, the dimension of latent space is more than the input, so it means the model has more capacity to represent input dara, and it may lead to overfitting.
Its beneficial fore:

- Learning more detail and representation of data
- Improves reconstruction quality

The disadvantages of these methods are

- Overfitting
- Computationally expensive

Exactly complete AE's: dimensions of input and latent are same. So it means it has no benefits than the overcomplete scenario(almost same) and it is also prone to be overfit!
And it's also the same as identity mapping.

Q2-

The training process of the AE's is different from traditional supervised learning with labels!

In the supervised techniques, the model takes an input and makes an output, which then compares with the *Truth* value of that input record and it leads to a loss value, which then backpropagates through the network. But in AE's, which is an unsupervised technique, we don't have any Truth value and label, but we also need a criterion and a measure to define the goodness of our model! In these cases, like AE's, we define different learning approaches and losses to overcome this issue. For example, in AE's we construct a new image based on the output of the latent layer, and also we want our output to be something!!(denoised image, reduced dimensions etc.). So we compare the constructed image with the original image, which is known as *reconstruction error,* then we can backpropagate this error for the training! And it's how we can train an AE to reconstruct an image.

Q3-

The reconstruction loss which is used in the AE's is expected to have a negative log likelihood of the reconstructed image distribution with respect to encoder distribution. Which is as follow:

$$l_i(\theta, \phi) = -E_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)]$$

q-theta: encoders distribution, p-phi = representation
It means that, with the latent variable Z, what is the likelihood of the representation to be the same as the input data, which is quite logical for the reconstruction loss part. On other hand, we can use **MSE** as reconstruction loss as well but the result would not have sufficient quality since the MSE is not a robust loss module and it leads to a blur representation. In case of **KL divergence**, it's more like a regularizer for AE's since it force the distribution of representation to a specific distribution like standard normal distribution, and it can't be as a reconstruction loss, the formula is as follows:

$$D_{\mathrm{KL}}(p(x) \,||\, q(x)) = \sum_{x \in X} p(x) ln \quad \frac{p(x)}{q(x)}$$

It measures the difference between two probability distributions.

And also, **BCE** is used when the input data and reconstructed output are binary or probabilistic, so it's not a general purpose loss function for AE's.

Q4-

There are several ways to evaluate the quality of learned representations in AEs, But it's more challenging than traditional supervised learning evaluations like MSE, BCE and etc. here is some approach to evaluate these models:

1- Reconstruction error: we discussed this metric in **Q3**, which compares the reconstructed output with the original data.

2- KL divergence: it's not exactly a metric for evaluating the learned representations, but it can lead to a better latent space and more interpretable space.

3- Cluster analysis: the clustering methods like k-means could apply on latent space, if the latent is meaningful, datapoints which are close together are clustered in the same cluster, indicating good separation of data.

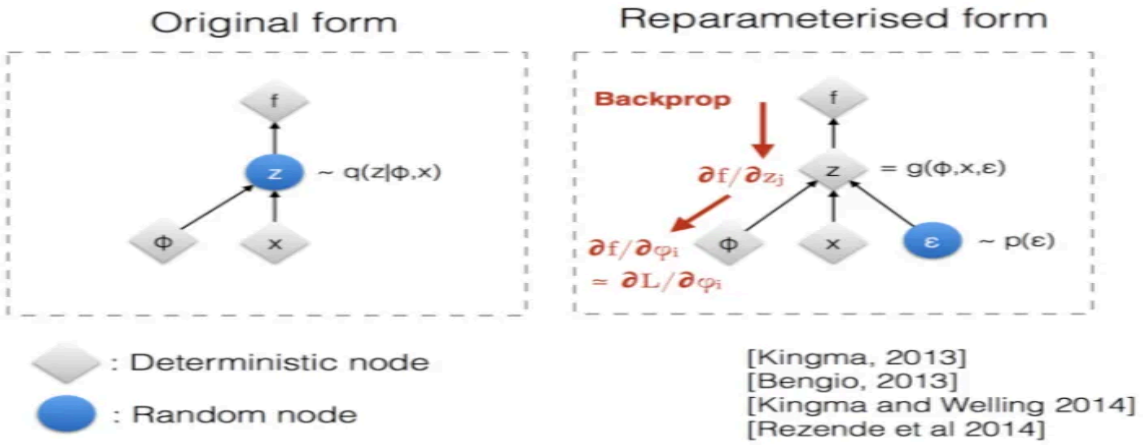
4- Reconstruction visualization: by visualizing the reconstructed output and comparing it to input data, we can capture this fact of how good our model could represent the data and fully understand the features.

5- Contrastive learning: Contrastive learning methods, such as SimCLR or MoCo, can be used to evaluate the quality of representations by training the model to discriminate between similar and dissimilar pairs of data points in the latent space. A higher contrastive loss indicates that the representations are semantically meaningful.

Q5-

In concept of the VAE's, The sampling process is a stochastic process which is not differentiable and the gradient could not backpropagate in the network, so the reparametrization technique is to allow applying a gradient based optimization.
But why can't we backpropagate the gradient from models involving stochastic variables?

By introducing randomness, sampling poses some challenges in the gradient-based optimization because non-differentiable operations, like sampling, prevent the computation of gradients. The reparameterization trick addresses this issue by sampling from a standard distribution and then transforming the sample in a deterministic manner, which allows for gradient-based optimization, making the transformation differentiable regarding the parameters of the target distribution. This involves sampling ε from a standard Gaussian instead of sampling directly from the target distribution specified by its mean μ and the covariance matrix Σ, allowing backpropagation through random nodes of VAEs.



$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I})$$
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$$

Q6-

The latent space of a Variational AutoEncoder is regularized to favor the emergence of disentangled and interpretable representations. Here are some ways this can be done:

1. Beta Variational AutoEncoder (β-VAE): Add a hyperparameter β to modulate the strength of the regularization term in the VAE objective function. Increasing β will push the model to learn more disentangled representations

2. L1 or L2 Regularization: Apply L1 or L2 regularization over the latent variables to encourage either sparse or smooth representations, respectively.

3. Orthogonality Regularization: Regularize the latent space to have orthogonal dimensions, which can lead to more disentangled representations
4. FactorVAE: Use a factorial prior distribution in the VAE, encouraging the latent variables to be independent and disentangled.
5. Clustering-based Regularization: Regularize the latent space to cluster similar data points together, hence promoting more disentangled and interpretable representations.

I want to explain more about the clustering method since i found it interesting:

By applying the clustering on the latent space, it leads to separation of factors, also it leads to more structures latent space when the similar data points are close together and dissimilar points are far apart, and it makes the latent space more interpretable. And also we can use this method to reduce the dimensionality of the latent space by keeping only important features in this layer.

Q7-

Three main challenges we face while using the high-dimensional and input distributions datasets:

1- Overfitting: high dimensional inputs can lead to overfitting since the data could be noisy and the model could not generalize the data.
2- Mode collapse: this problem occurs when the model produces limited variations of the same output, rather than exploring the possible range of output.
3- Posterior collapse: where the posterior distribution becomes identical to the prior distribution and the model just learns to ignore the input data.

We can overcome above issues using these techniques:

- **Dimension reduction:** techniques like PCA and t-SNE are commonly used for this purpose.
- **Regularization:** apply techniques like L1 and L2 to avoid overfitting and encourage sparse representation.

- **Batch normalization:** for improving robustness and stability of the model.
- **Warm-up:** apply the warm up phase during training to help models learn good representation before latent space becomes entangled.

And etc.

Q9-

Here is some of different convolution layer types:

1- regular conv layer: applies a simple kernel on data to extract features. It is used in image, video and time series processing. For example a simple 3*3 conv kernel used in common CNN networks.
2- Dilated conv layer: expands the receptive field of conv kernels by adding dilations between coefficients. It allows the filter to capture a wide context of the input data without increasing the parameters.(we will discuss this type of conv layers in next question)
3- transposed conv layers: it performs the reverse operation on the output of a conv layer for upsampling and reconstructing an image from the lower-resolution representation.
It is used in scenarios like upsampling tasks like image super-resolution and generation, and semantic segmentation. (we used them in the previous series as well)
4- Group conv: splits the input channels into the groups and apply conv filter on each group, it's more efficient and reduces the number of parameters.
So it is used in common networks with any possible tasks, but when we need computational efficiency.
5- Deformable convs: it makes conv kernels to be deformed, leading to make conv operation more flexible to different input patterns. It is used in tasks like object detection and image segmentation when we need to capture complex shapes and deformations.

6- Depthwise separable conv: it divides conv layer into two separable operations: 1- depthwise and 2- pointwise convolution.

Depthwise applies a filter for each input channel separately, pointwise combining the outputs of the depthwise. It is used in scenarios like feature extraction for mobile vision tasks and scenarios when the computational resources are limited.

7- Spatial transformer networks: applies affine transformation on input image, allowing the network to focus on important parts of the image.
Used in image classification, obj detection and image captioning tasks.

Q10-

Dilated convolution is a type of conv layers which allows the network to increase receptive fields without increasing the number of parameters.
This could be achieved by adding gaps(dilation) in the kernel. The main benefits of these layers is capturing the log-range dependencies from the input(image, signal etc.) let see how it actually works.

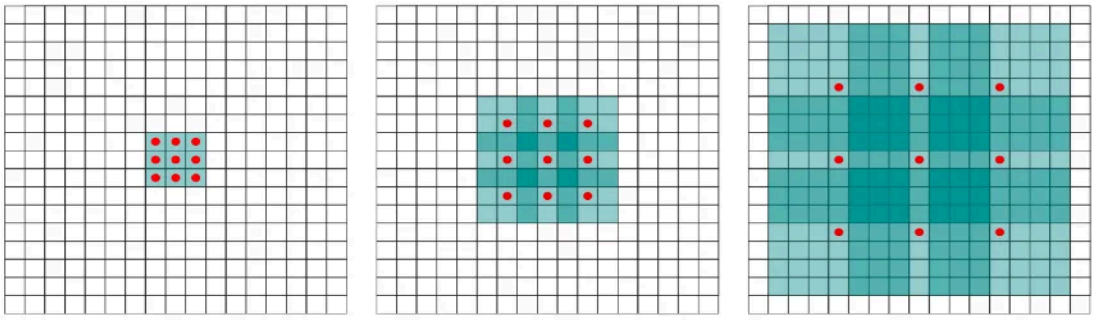


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

As you can see in the figure above, by adding gaps in the kernels, the receptive fields are growing exponentially without loss of resolution and coverage.
According to these, the dilation convs are used in tasks which need a more global view of an input data:
image segmentation tasks, to be more specific, **medical image segmentation**, which is needed to be more accurate for diagnosis purposes.

Autonomous driving, in this task, dilated conv can help to improve obj detection and *scene understanding* by increasing the global view.
Satellite imagery analysis, dilated conv could be helpful for large-scale pattern recognition and capture the relationships between different features such as land use, vegetation, and water bodies.

Q8-

Variational Autoencoders (VAEs) are powerful generative models, and their flexibility allows for various extensions to incorporate additional constraints or objectives, leading to wider practical applications. Here are a few key extensions and their benefits:

1. **Conditional Generation:**
   - **Concept:** Instead of learning a latent space for any data point, we can guide the VAE to generate outputs conditioned on specific attributes or labels. This is achieved by feeding the desired condition as an additional input to both the encoder and decoder.
   - **Implementation**: Concatenate the condition vector with the input vector in the encoder and the latent vector in the decoder.
   - **Benefits**:
     - Controllable Generation: Generate data with specific features (e.g., images of faces with different hairstyles).
     - Data Augmentation: Create synthetic data for underrepresented classes in a dataset.
   - **Example**: Generating images of handwritten digits conditioned on the specific digit (0-9).

2. **Semi-Supervised Learning:**

   - **Concept:**Leverage both labeled and unlabeled data to improve the model's performance. The VAE learns to reconstruct the input data and predict the labels for labeled data points.

- **Implementation:** Add a classification head to the encoder, predicting the label from the latent representation. The loss function combines the reconstruction loss (standard VAE loss) and the classification loss.
- **Benefits:**
    - Improved Performance with Limited Labels: Achieve higher accuracy with fewer labeled data points compared to purely supervised methods.
- **Example:** Training an image classifier with limited labeled images by leveraging a large dataset of unlabeled images.