AML questions

Lesson 2

1. **Generative vs Discriminative Learning**
   - Generative learning models learn the joint probability distribution $P(Y, X)$ of both the input features X and the output Y. They can be used to generate new samples from the input distribution.
   - Discriminative learning models learn the conditional probability distribution $P(Y|X)$ of the output Y given the input features X. They directly learn how likely a given input is to belong to a particular output class.
2. **Maximum A Posteriori (MAP) Estimation**
   - MAP seeks to find the value of a parameter that maximizes the posterior probability distribution, taking into account both the likelihood of the data and the prior distribution.
3. **Maximum Likelihood Estimation (MLE)**
   - MLE seeks to find the value of a parameter that maximizes the likelihood of the observed data, assuming a particular probability distribution. For large amounts of data, we can approximate MAP by MLE and use it to approximate our parameters.

Lesson 3

1. **Loss Functions**
   - We define the loss $L(h(X), Y)$ of the classifier $h(X)$ the function that measures the error between the assigned predictions and the true labels Y.
2. **True Risk**
   - The True Risk of a classifier h is the probability that it does not predict the correct label for a given input X, drawn from the distribution D.

$$R_D(h) = P(h(X) \neq Y)$$

   - We can also see the True Risk of a classifier as the **Expected Value** of the loss function over the distribution of the data.
3. **Bayes Risk vs Empirical Risk**
   - **Bayes risk** is the expected loss (True Risk) of a bayes classifier over the current distribution of the data. It is the best possible performance that can be achieved for a given problem and cannot be improved.
   - **Empirical risk** $R_D(h)$ over the distribution D is the average loss of the classifier over the training data. It is an estimate of the True Risk. It's impossible to obtain $R_D(h) = 0$ even with a Bayes Classifier. The predictions of any model are only approximately correct, so we'll allow our model to fail with a probability $\delta \in (0, 1)$.
4. **PAC Learnability**
   - An Hypothesis Class H is PAC-learnable if there exists a function m_H that, given enough samples, produces (with a certain probability) an hypothesis that can achieve a true risk $R_D(h) \leq \epsilon$.
   - Directly from the definition of Empirical Risk, we can see that a model is PAC-learnable if the empirical risk is close to the true risk ($R_D(h) \leq \epsilon$) with a certain probability $(1 - \delta)$. So our

model should be Probably (with probability $1-\delta$) Approximately (with error ϵ) Correct.
    - The number of samples m_H needed to achieve this is called the **sample complexity of the model**.
5. **Overfitting and Underfitting**
    - Overfitting is a phenomenon that occurs when a model is **too complex** or flexible and fits the training data too closely (including noise), resulting in poor performance on unseen data (test data) due to the model's inability to generalize.
    - Underfitting is a phenomenon that occurs when a model is **too simple** to capture the underlying patterns in the data. As a result it performs poorly on both the training and unseen data.
6. **K-Fold Cross Validation**
    - K-Fold Cross Validation is a technique used to evaluate the performance of a machine learning model. The training data is divided into K subsets, and the model is trained K times, each time using K-1 subsets for training and the remaining subset for validation. The performance of the model is then averaged over the K runs. If K is equal to the number of samples, it is called Leave-One-Out Cross Validation.
7. **Linear Regression Algorithms**

Regression algorithms are used to predict a **continuous** output variable based on the input features. In classification $Y = 1, 2, 3, \ldots, K$, in regression $Y \in \mathbb{R}$.

- **Loss Functions** for regression are typically based on the difference between the predicted value and the true value. A common loss function is the Mean Squared Error (MSE) $L(y, y) = (y - y)^2$.
- **Linear Regression** is a simple and commonly used regression algorithm that models the relationship between the input features and the output variable as a linear function.
- **Ridge Regression** is a variation of linear regression that adds a regularization term to the cost function, to prevent overfitting.

Lesson 4

1. **Logistic Regression** is a non-linear regression algorithm that is used for binary classification. It models the probability that an input belongs to a particular class using the logistic function. The loss function is :

$$l(x_i, y_i) = \sum_{i=1}^{n} (y_i - \sigma(ax_i + b))^2$$

We can rewrite the loss function to have a convex problem as :

$$l(x_i, y_i) = \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i \ln(1 - \sigma(ax_i + b)))$$

But that function still has no analytical solution, so we use the **Gradient Descent** algorithm to minimize the loss function.

2. **Gradient Descent** is a first-order algorithm used to minimize a loss function by **iteratively updating** the parameters of the model in the direction of the negative gradient of the loss function : $x^{(t+1)} = x^{(t)} - \alpha \nabla f(x^{(t)})$ and is orthogonal to level surfaces. It requires that the function f is **differentiable** at all points.

- **Stationary point** is a point where $\alpha \nabla f(x^{(t)}) = 0$ so the algorithm "remain stuck" at that point. We have no guarantee that the point is a minimum, it could be a saddle point for example. On the other hand in DL we're not interested in the global minimum because it would cause overfitting.
- The **Learning Rate** α is a hyperparameter that controls the size of the updates. If the learning rate is too small, the algorithm may take a long time to converge, while if it is too large, the algorithm may oscillate or diverge. The size of the step is given by $\alpha||\nabla f||$. The learning rate can be fixed, adaptive or follow a schedule.
- *decay* is a technique used to reduce the learning rate over time, which can help the algorithm to converge more effectively.
- *momentum* is a technique used to accelerate the convergence of the algorithm by adding a fraction of the previous update to the current update.
- Two major bottlenecks of the algorithm make it unsuitable for Deep Learning: the number of samples (we're computing the gradient over the entire dataset) and the number of parameters (milions of parameter!).

3. **Stochastic Gradient Descent (SGD)** is a variant of gradient descent that updates the parameters of the model using a **single batch of data** B at a time, rather than the entire training set. This can lead to faster convergence, but it can also be more noisy and may require more iterations to converge. Small batches can also offer a regularization effect. Compared to GD, SGD can find low value of the loss quickly enough to be useful in DL, even if most of the time it's not the minimum.

Lesson 6

1. **The perceptron** is a simple classifier loosely inspired by the way neurons work in the brain. It is used for binary classification and is based on the product of the **input features** and the **weights**, which are then summed and passed through an **activation function**. If the sum is greater than a threshold, the perceptron outputs +1, otherwise it outputs -1. To train the perceptron, it computes one sample at a time,if the sample is correctly classified the weights remain the same, otherwise the weights are updated by adding/removing the sample(feature vector) to the weights. If the data are linearly separable, the perceptron algorithm is guaranteed to converge to a solution. If the data are not linearly separable, the perceptron algorithm may not converge.

2. **Multilayer Perceptron (MLP)** is a type of neural network that consists of multiple layers of **neurons**, including an input layer, one or more hidden layers, and an output layer. Each neuron in the network is connected to every neuron in the adjacent layers, and each connection has a weight associated with it. The network is trained using **backpropagation** of gradient.

3. **Why non linearity is important in cnn?**

    - Non-linearity is important in CNN because it allows the network to learn more complex representations of the data. This is because linear functions can only represent linear relationships, but non-linear functions can represent more complex relationships.

Lesson 7

1. **Convolutional Neural Networks (CNNs)** are a type of neural network that is well-suited for tasks such as image recognition and classification. They use a special type of layer called a **convolutional layer**, which applies a set of filters (kernel) to the input data to extract features. CNNs also use other types of

layers such as pooling layers and fully connected layers to further process the data and make predictions.

- **Convolutional Layer** is a layer that **convolves** the input data with a set of filters to extract features. This can help the network to learn to recognize patterns in the input data. The output size of the convolutional layer is given by the formula $O = \frac{N-F+2P}{S} + 1$ where N is the input size, F is the kernel size, P is the padding and S is the stride. Each filter in the convolutional layer can learn to recognize different patterns and features in the input data.
- **Pooling Layer** is a layer that reduces the spatial dimensions of the input data by combining nearby values. This can help to reduce the computational cost of the network and make it more robust to small variations in the input data. An example of pooling is the Max Pooling layer.
- **Fully Connected Layer** is a layer that connects every neuron in one layer to every neuron in the next layer. This is the final layer of the network and is used to make predictions.

Lesson 8

1. **Batch Normalization** is a technique used in deep networks to **normalize** (zero mean and unit variance) the output of a layer before it is passed to the next layer. Note that this is a **differentiable function**, so it can be trained with backpropagation. The params of the layer (γ and β) are learned during training rather than picked as hyperparameters. Note that at test time the BatchNormalization layer will act differently, using a fixed mean and variance computed during training. Usually inserted after Fully Connected or Convolutional layers and before the activation function (non-linearity). Batch Normalization can help to:

    - Optimize the **Internal Covariate Shift** that is the dynamic change of weights and biases of the network during training. An Internal Covariance Shift too big can slow down the training of the network.
    - Improve gradient flow through the network, which can help to speed up training and reduce the risk of vanishing or exploding gradients.
    - Reduces the dependence of the network on the initial values of the weights and biases, which can make training more stable and less sensitive to the choice of hyperparameters.
    - Acts as a form of **regularization** by adding noise to the input of the layer, which can help to prevent overfitting.

2. **Activation Functions** are used to introduce non-linearity into the output of a neuron. There are different possible activation functions that can be used in neural networks, such as the sigmoid, tanh, ReLU, and softmax functions.

    - **Sigmoid** is a non-linear activation function that squashes the input to the range [0, 1]. Has three main problems: the vanishing gradient problem, the output is not zero-centered and the exp() function is computationally expensive.
    - **Tanh** is a non-linear activation function that squashes the input to the range [-1, 1]. It's zero-centered but still has the vanishing gradient problem.
    - **ReLU** (Rectified Linear Unit) is a non-linear activation function that returns the input if it is positive, and zero otherwise. It is commonly used in Neural Networks because it is computationally efficient and does not suffer from the vanishing gradient problem. However, it can suffer from the "dying ReLU" problem, where neurons can become inactive and stop learning, and has a non-zero centered output.

- **Leaky ReLU** is a variation of the ReLU activation function that allows a small gradient when the input is negative, which can help to prevent the "dying ReLU" problem.
- **ELU** (Exponential Linear Unit) is a non-linear activation function that returns the input if it is positive, and an exponential function of the input otherwise. It is similar to the ReLU function, has an output closer to zero-mean but is computationally expensive.

The best choice of activation function remains the ReLU function, using a Leaky ReLU or ELU function can improve just a little bit the performance of the network.

3. **Weights Initialization** is an important aspect of training neural networks. The initial values of the weights can have a significant impact on the performance of the network.

    - **Zero Initialization** is a method for initializing the weights of a neural network that sets all the weights to zero. Not recommended because it can lead all the layers to evolve in the same way loosing the specialization.
    - **Random Initialization** sets all weights to random values. This can help to break the symmetry of the network but works only on small networks, on deeper networks can couse problems with the gradient.
    - **Xavier Initialization** is a weights initialization method for a neural network that is designed to keep the **variance** of the activations constant across layers. It is based on the assumption that the input and output of each layer are normally distributed. The Xavier initialization method scales the weights by a factor of $\sqrt{\frac{1}{n_{in}}}$ where n_{in} is the number of input units to the layer. It works only with the tanh activation functions.
    - **MSRA Initialization** is a variation of Xavier initialization that is designed to work better with the ReLU activation function. It scales the weights by a factor of $\sqrt{\frac{2}{n_{in}}}$ where n_{in} is the number of input units to the layer.

4. **Dropout** is a regularization technique used in neural networks to prevent overfitting. It works by randomly setting a fraction of the input units to zero at each update during training. The probability of dropping a unit is a hyperparameter that can be tuned (usually 0.5). Forces the network to have redundant representation and thus better generalization. It can also be seen as the training of an ensemble of networks. Usually done in Fully Connected layers to lower the complexity. At test time all neurons will be active but their output will be scaled by the dropout probability (or divide at test time).

5. **Data Augmentation** is a technique used to increase the amount of data available at training time by applying transformations to the input data. This can help to improve the performance of the network and reduce the risk of overfitting. Common data augmentation techniques include flipping, rotating, scaling, and cropping the input data.

6. **Model Ensembling** is a technique used to improve the performance of a machine learning model by training independent models and combining their predictions. At test time average the predictions of the models. Small increase in performance and it's computationally expensive.

Lesson 9

1. **Hyperparameter Tuning** is the process of finding the best set of hyperparameters for a machine learning model. This can be done using techniques such as **grid search** or random search. The best set of hyperparameters is the one that results in the best performance on a validation set.

2. **Transfer Learning and Fine Tuning**

    - Transfer learning is a technique where a pre-trained model is used as a starting point for a new task, rather than training a model from scratch. This can save time and resources, and also improve performance. With Transfer Learning, the weights of the pre-trained model are frozen, and only the weights of the last layer are trained on the new task. An example is using a model pre-trained on ImageNet to perform image classification on a new dataset, changing only the last layer.
    - Fine-tuning is a similar technique where the weights of the pre-trained model (more layers) are updated for the new task, typically by using a smaller learning rate. The number of layers that we want to re-learn depends on the size of the new dataset and the similarity between the new and the old dataset.

3. **AlexNet** is the first Deep Convolutional Neural Network that won the ImageNet competition in 2012. It is composed of 5 convolutional layers, 3 max-pooling layers, and 3 fully connected layers. It uses the ReLU activation function and dropout for regularization. It also uses data augmentation and batch normalization to improve the performance of the network.

Lesson 10

1. **GoogleNet**

    - GoogleNet is a deep convolutional neural network that was designed to be computationally efficient (no Fully Connected layers) while achieving high performance on image classification tasks. It uses a module called an **Inception module** that combines multiple convolutional layers with different kernel sizes and pooling layers to extract features from the input data. The network also uses **global average pooling** instead of Fully Connected layers to reduce the spatial dimensions of the input data before making predictions. The network is too deep to be trained only with loss at the end, so it uses two **auxiliary classifiers** to help the training of the network.

2. **Residual Networks (ResNets)**

    - ResNets are a type of deep convolutional neural network that use a special type of layer called a **residual block**. A residual block consists of a set of convolutional layers followed by a **skip connection** that adds the input to the output of the convolutional layers. This allows the network to learn the residual (difference) between the input and the output, which can help to prevent the vanishing gradient problem and make it easier to train very deep networks. The skip connection can be implemented in different ways, such as adding the input to the output.

3. **Neural Architecture Search (NAS)**

    - Neural Architecture Search is a technique used to automatically design the architecture of a neural network. One controller outputs network architectures that are then trained and evaluated. After training a bunch of models, make a gradient step on the controller. Over time the controller model will learn to generate better architectures. The search space is really large and the training of the models is computationally expensive.

Lesson 11

1. **Multitask Learning**

- Multitask learning is a technique where a model is trained to perform multiple tasks simultaneously. This can help to improve the performance of the model by allowing it to learn shared representations of the data that can be used for multiple tasks. The model is trained on a joint loss function that combines the losses of the individual tasks.
- Multi-task vs Transfer Learning: In multi-task learning, the model is trained to perform multiple tasks simultaneously, while in transfer learning, a pre-trained model is fine-tuned for a new task.

2. **Object Detection** is the task of identifying and localizing (multiple) objects within an image or video. Various techniques can be used for object detection, such as sliding window, region-based, and single-shot detection methods.

    - **Sliding Window** Apply a CNN to many different crops of the image, CNN classifies each crop as object or background. The main problem is the computational cost (an 800x600 images has 58M boxes).
    - **R-CNN** Use a **region-proposal** (selective search) algorithm to propose a set of regions with "blobby" shapes, then the CNN is applied to each region. The main problem now is the selective search algorithm that is slow and produces too many regions.
    - **Fast R-CNN** Runs th whole image through the Backbone CNN to extract a feature map of the image, then selects the RoIs with a proposal method and applies smaller CNNs to each RoI. The main problem is how to crop and resize the RoIs to match the original image (RoI Pool vs RoI Align).
    - **Faster R-CNN** Uses a Region Proposal Network (RPN) after the backbone to generate the RoIs, then applies the Fast R-CNN to each RoI. The RPN is a small CNN that takes the feature map of the image and outputs the RoIs.
    - **Single Stage Detector** (SSD, YOLO) are object detection algorithms that use a single network to predict the bounding boxes and class labels for all objects in an image. They are faster than two-stage detectors but may have lower accuracy.

3. **Intersection over Union (IoU)** is a metric used to evaluate the performance of object detection algorithms. It measures the overlap between the predicted bounding box and the ground truth bounding box. The IoU is calculated as the **area of intersection** between the two bounding boxes divided by the **area of union** between the two bounding boxes.

4. **Non-Maximum Suppression (NMS)** is a technique used to reduce the number of overlapping bounding boxes produced by an object detection algorithm. It works by selecting the bounding box with the highest confidence score and removing any other bounding boxes that have a high overlap with it. This can help to improve the precision of the object detection algorithm.

5. **RoI Pool vs RoI Align**

    - RoI Pooling is a technique used in object detection algorithms to extract features from a region of interest (RoI) in an image. It works by dividing the RoI into a grid of sub-windows and then "snapping" the features to the grid. This can lead to a loss of information and reduce the accuracy of the object detection algorithm.
    - RoI Align is a variation of RoI pooling that uses **bilinear interpolation** to align the features with the grid. This can help to preserve the spatial information and improve the accuracy of the object detection algorithm.

6. **Semantic Segmentation** is the task of classifying each pixel in an image according to the object it belongs to. It is a pixel-level classification task that can be used to create detailed and accurate object masks. Various techniques can be used for semantic segmentation, such as fully convolutional networks and U-Net.

7. **Fully Convolutional Networks (FCNs)** are used to perform semantic segmentation by applying **Downsampling** and **Upsampling** inside the network. First a series of convolutional layers are applied to the input image to extract the **features vector**. The network is then designed to upsample the feature vector and the obtained output is a **pixel-wise** classification with original image size C x H x W where C is the number of classes.

    - **Downsampling** is performed by applying a series of convolutional layers alternated to Pooling layers. The convolutional layers are used to extract the features from the input image, while the pooling layers are used to reduce the spatial dimensions of the feature map.
    - **Upsampling** is performed by applying "Unpooling" layers like Bilinear Interpolation, Max Unpooling etc. No learning is involved in this types of layers. An alternative is the **Transposed Convolution** that is a learnable layer that can be used to upsample the feature map.

8. **Instance Segmentation** is the task of identifying and localizing each object instance in an image. It is a combination of object detection and semantic segmentation, and can be used to create detailed and accurate object masks for each object in an image. Can be performed with a **Mask R-CNN** that extends the Faster R-CNN by adding a new task after RoI pooling : **Mask Prediction**.

9. **Panoptic Segmentation** is the task of simultaneously performing semantic segmentation and instance segmentation on an image. It can be used to create detailed and accurate object masks for each object in an image, as well as to classify each pixel according to the object it belongs to.

Lesson 12

1. **Sequential Data** are informations that are not related to a single instant, but are extended in time. Examples can be Music, Text, Stock Exchange, Weather, etc.

2. **Different Tipes of Networks**

    - **One-to-One** : "vanilla" networks, takes a single input and produces a single output.
    - **One-to-Many** : Takes a single input and produces multiple outputs. An example can be Image Captioning, where the network takes a single image in input and produces multiple words.
    - **Many-to-One** : Takes multiple inputs and produces a single output. An example can be Sentiment Analysis, where the network takes multiple words in input and produces a single sentiment.
    - **Many-to-Many** : Takes multiple inputs and produces multiple outputs. An example can be Machine Translation, where the network takes multiple words in input and produces multiple words in output.

3. **(vanilla) Recurrent Neural Networks (RNNs)** are a type of neural networks that have an **internal state** that is updated as a sequence is processed. The new state of the network is given by both the old state and the input. In "vanilla" RNNs we have a **single set of weights** that is used at every time step, so the same parameters must work with all the elements.

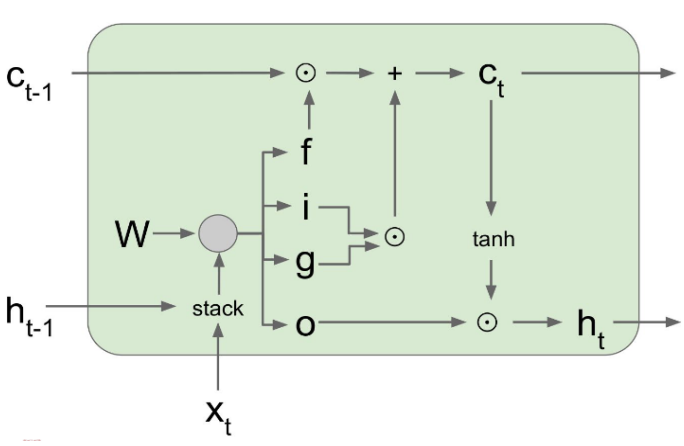
- **Backpropagation Through Time (BPTT)** is a technique used to train RNNs by unrolling the **whole network** through time and applying backpropagation to update the weights. The main problem is the vanishing gradient problem, that can make the training of the network really slow and unstable for bigger texts.
- **Truncated BPTT** is a variation of BPTT where the text is truncated into parts (chunks) and the weights are updated at the end of each chunk. This can help to reduce the computational cost of training the network.

4. **(vanilla) RNNs problems** are related to the use of a single weight matrix for all the time steps. The main problems are :

   - **Vanishing Gradient** : If the largest singular value of the weight matrix is less than 1, the gradient will vanish over time. Can be solved only by changing the architecture of the network.
   - **Exploding gradient** : If the largest singular value is greater than 1, the gradient will explode over time. Can be solved with **gradient clipping**.

5. **Long Short-Term Memory (LSTM) Networks** are an extension of vanilla RNNs used to solve the vanishing gradient problem. In LSTM (other than input and state) we have also a **cell state** that won't be exposed to the outside world. Solves the vanishing gradient problem because now we have two possible paths for the gradient to flow (the cell state and the hidden state) obtaining in this way an uninterrupted gradient flow.

   - $c_t = f * c_{t-1} + i * g$: The cell state is updated by a combination of the old cell state and the new input. f is the forget gate, i is the input gate and g is the "write" gate.
   - $h_t = o * tanh(c_t)$: The hidden state is updated by the cell state and the output gate. o is the output gate.



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

here we're deciding what to forget — here decide to add a value +1

internal state $c_t = f \odot c_{t-1} + i \odot g$

cell state at time t-1

$h_t = o \odot \tanh(c_t)$ — we pass the internal state to the tanh

output gate
sigmoid function
how much we want to reveal about the inside state

6. **Attention** is a mechanism used in neural networks to selectively focus on certain parts of the input when making a prediction. This allows the model to attend to the most relevant information in the input, and is particularly useful in tasks such as image and text understanding.

7. **Self Attention Layer** is a type of attention mechanism that is applied to the input data within a single layer of a neural network. This allows the model to learn to attend to different parts of the input data within the same layer, rather than having to pass the data through multiple layers.

8. **RNN with Attention** consists of an **encoder** that processes the input sequence and produces a sequence of hidden states.

    1. The last hidden state s_0 is combined with each hidden state through a Fully Connected Layer (MLP) to compute the **alignment scores** $e_{t,i} = f_{att}(s_{t-1}, h_i)$.
    2. The alignment scores are then passed through a **softmax** function to obtain the **attention weights** $a_{t,i}$.
    3. The attention weights are then used to compute the **context vector** by taking a weighted sum of the hidden states. $c_t = \sum_i a_{t,i} h_i$

9. **Advantages of RNN with Attention** are that the whole network is differentiable, so it can be easily trained with backpropagation, and that the attention mechanism allows the model to focus on the most relevant parts of the input data. Note that at each time step the context vector is different, so the model can focus on different parts of the input at different time steps.

Lesson 13

1. **Domain Shift** is the change on data distributions when we move from a dataset to another. Bigger the domain shift, bigger the performance drop of the model because we assume for each model that the data are I.I.D (Indipendent and Identically Distributed). In Images we can have domain shifts due to different lighting conditions, different camera angles, different backgrounds, etc. In Text we can have domain shifts due to different writing styles, different languages, different topics, etc.

2. **Domain Adaptation** is a technique used to adapt a model trained on one dataset to work well on a different but related dataset. Note that we're assuming that the model will do the **same task** on both dataset.

3. **Adversarial Domain Adaptation (RevGrad)** is a Domain Adaptation technique that aims to make the two distributions (source and target) as close as possible. In theory, once the two distributions are completely overlapped we can train the model on the source domain and use it on target domain without performance degradation. In **RevGrad** we have a **domain classifier**, added to the base model, that tries to distinguish between the source and target domain. To overlap the to domains we'll **reverse the gradient** of the domain classifier, so the domain classifier will try to distinguish between the two domains and the base model will try to make the two domains as close as possible.

4. **ADDA** (Adversarial Discriminative Domain Adaptation) is a method for unsupervised domain adaptation, it is an extension of GANs, where the generator network is trained to generate samples from the target domain, and the discriminator network is trained to distinguish between samples from the source and target domains.

5. **Cycle GAN** is a type of GAN that is able to translate images from one domain to another, it is trained by using cycle consistency loss which trains the model to maintain the same content and style of the input image after translation.

6. **SBADA-GAN** is a semi-supervised generative adversarial network (GAN) that is used for cross-domain adaptation. It is designed to adapt a model trained on one dataset to work well on a different but related dataset.

7. **Pixel level DA** is a technique used to adapt a model at the pixel level. This is used when the input images have different characteristics such as different lighting conditions, resolutions or image noise.

Lesson 14

1. **Supervised Learning** : Based on the use of labelled data, the goal is to learn a function that maps the input to the output labels. Classification, regression and semantic segmentation are examples of supervised learning. In some cases is really costly to obtain labelled data and errors in the labelling can lead to poor performance and noise.

2. **Unsupervised Learning** : Our model does not have access to labels, all work is note based only on data, Generatives models are an example of unsupervised learning. Density estimation is also an example of unsupervised learning where the goal is to estimate the probability density function of the data. The main goal is to find some **underlying structure** in the data.

3. **Self-supervised Learning** : Type of unsupervised learning where we try to extract informations from the data in a way similar on how kids learn: by observing the world, interacting with stuff and see how it reacts. Usually we take the data, remove some informations and then task the model to predict the removed informations. The main goal is to learn a good representation of the data. And example of Self-supervised learning is to extract two patches from an image and then provide the model with two different patches and task it to predict the relative position of the two. In this way we're able to train on huge amount of data without the need of labels and our model will try to find some underlying structure of the data.

4. **Contrastive Learning** : Another type of unsupervised learning, here we start from a set of samples and extend them with data augmentation (removing colors, jittering, cropping etc) and then put the augmented data through multiple networks. When two patches from the same image are taken, we expect them to be classified as close as possible, while when two patches from different images are taken, we expect them to be classified as far as possible.

5. **Vae vs GAN** VAE (Variational Autoencoder) and GAN (Generative Adversarial Network) are both generative models, but they work differently. VAE is a probabilistic model that aims to learn a latent representation of the data, while GANs are trained to generate new data by pitting a generator network against a discriminator network. Specifically :

    - VAEs use deep learning techniques to learn a latent representation (z) of input data (x) and are based on **AutoEncoders**, which are neural networks designed to learn a compressed representation of the input data, but with VAEs the representation is **probabilistic** rather than deterministic.
    - GANs are based on two neural networks playing a "MiniMax" game. The **Generator** creates new data samples with the intention of fooling the discriminator, while the discriminator tries to distinguish between the generated samples and real samples. There are no explicit likelihood distributions in GANs. Instead of minimizing the probability that the discriminator is right (gradient descent), we can maximize the probability that the discriminator is wrong (gradient ascent). The gradient works much better this way. Training the two networks together is challenging and can be unstable even with this optimization.

General notes

Softmax : is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes. The function takes as input a vector z of K

real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers, indeed map the interval of values to the interval [0,1] (exponential)

Cross Entropy loss: is a loss function used to calculate the loss between probabilities, $H(P,Q) = -\sum_{i=1}^{N} P(i)log(Q(i))$, where P is the true probability of the class/event and Q is the predicted probability of the class/event. The cross-entropy loss is a measure of how different your predicted distribution is from the true distribution of the data.

Top-N error : is a metric used to evaluate the performance of a classification model, it is the percentage of the time that the true class is not in the top N predicted classes. For example, if the top-5 error is 0.15, that means that 15% of the time the true class is not in the top 5 predicted classes (the top-5 most probable classes do not include the true class)

Other AML questions

1. **batch normalization**
    - Batch normalization is a technique used to normalize the input layer by adjusting and scaling the activations. It helps to reduce internal covariate shift, which can improve the stability and speed of training.
2. **Gradient policy vs q learning**
    - Gradient policy and Q-learning are both reinforcement learning algorithms, but they work differently. Gradient policy is a type of policy-based algorithm that uses gradient descent to optimize the policy, while Q-learning is a value-based algorithm that estimates the value of a state or action.
3. **Self supervised learning**
    - Self-supervised learning is a type of machine learning where the model learns from input data without the need for explicit labels. It can be used to learn useful representations of the data that can be used for other tasks.
4. **Perceptron vs knn**
    - Perceptron and KNN (k-nearest neighbors) are both supervised learning algorithms, but they work differently. Perceptron is a simple algorithm that can be used for binary classification, while KNN is a non-parametric algorithm that is used for classification and regression.
5. **Generative vs discriminator**
    - Generative and discriminator models are both used in GANs. The generator model generates new data, while the discriminator model is trained to distinguish between real and generated data.
6. **Reinforcement learning**
    - Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The

goal of RL is to learn a policy, which is a strategy that specifies the action the agent should take under each possible state. The agent's objective is to maximize the cumulative reward over time.

7. **Domain adaptation**
    - Domain adaptation is a technique used to adapt a model trained on one dataset to work well on a different but related dataset.
8. **Back propagation**
    - Backpropagation is an algorithm used to train neural networks by updating the weights in the network so as to minimize the error of the output.
9. **How optimization is done in cnn**
    - In CNNs (Convolutional Neural Networks), optimization is typically done using methods such as stochastic gradient descent (SGD) or Adam.
10. **Semantic segmentation**
    - Semantic segmentation is a task where each pixel in an image is classified according to the object it belongs to.
11. **Instance segmentation**
    - Instance segmentation is similar to semantic segmentation, but it also involves separating individual instances of the same object class.
12. **Object detection**
    - Object detection is the task of identifying and localizing objects within an image or video.
13. **Multitask vs transfer learning**
    - Multitask learning is a technique where a model is trained to perform multiple tasks simultaneously, while transfer learning is a technique where a pre-trained model is fine-tuned for a new task.
14. **Regression vs classification**
    - Regression is a type of supervised learning where the goal is to predict a continuous output variable, while classification is a type of supervised learning where the goal is to predict a discrete output variable.
15. **Batch normalization and its use for domain adaptation**
    - Batch normalization can be used for domain adaptation by normalizing the activations of a pre-trained model to match the distribution of the new dataset.
16. **how to choose hyperparameters**
    - Hyperparameters can be chosen through methods such as grid search, random search, or Bayesian optimization.

17. **Why we initialize the weights in cnn**
    - In CNN, we initialize the weights to small random values to break symmetry and avoid zero gradients. This allows the network to learn different features in different layers.
18. **Layer freezing**
    - Layer freezing is a technique used to keep the weights of certain layers fixed during training. This can be useful when fine-tuning a pre-trained model, where we want to train only the last few layers.
19. **Deep learning, how to find the best parameters**
    - Finding the best parameters for deep learning can be done through techniques such as grid search, random search, and Bayesian optimization. These methods allow us to test different combinations of hyperparameters to find the best set that results in the best performance on a validation set.
20. **Shallow learning vs deep learning**
    - Shallow learning refers to traditional machine learning algorithms such as linear regression and decision trees, which have a small number of parameters. Deep learning, on the other hand, is a subfield of machine learning that uses neural networks with multiple layers, and a large number of parameters.
21. **Describe learning process**
    - The learning process in neural networks is the process of adjusting the weights of the network to minimize the error between the predicted output and the desired output. This is done through backpropagation and gradient descent.
22. **Pca vs autoencoder**
    - PCA (Principal Component Analysis) is a technique used for dimensionality reduction, it transforms the data into a new coordinate system where the dimensions are ordered by the amount of variance they explain. Autoencoder, on the other hand, is a neural network that aims to reconstruct its input, it is used for unsupervised feature learning and dimensionality reduction.
23. **Describe GAN**
    - GANs (Generative Adversarial Networks) are a type of deep learning model that consists of two parts: a generator network and a discriminator network. The generator network generates new data samples, while the discriminator network tries to distinguish between the generated samples and real samples.
24. **RevGrad**
    - RevGrad is a technique used to adapt a model trained on one domain to another domain. It works by reversing the gradient of the domain classifier and using it to update the model's parameters.
25. **batch normalization for DA**

- Batch normalization is a technique used to normalize the input data in order to improve the performance of a model. In the context of domain adaptation, batch normalization is used to adjust the model's internal representations to better align with the target domain.

26. **multi source BATCH NORMALIZATION for DA**

    - Multi-source batch normalization is an extension of batch normalization for domain adaptation, where the model is trained on multiple source domains and then adapted to a target domain. This allows the model to learn more robust features that are invariant to multiple source domains.

27. **Domain generalization**

    - Domain generalization is a technique used to train models that can generalize well to unseen domains. The goal is to learn features that are robust to variations in the input data, allowing the model to work well on unseen data.

28. **RNN-LSTM**

    - Recurrent Neural Networks (RNNs) are a type of neural network that can process sequential data such as time series or natural language. Long Short-Term Memory (LSTM) is a specific type of RNN that is designed to handle long-term dependencies in the input data.

29. **transformer**

    - Transformer is a type of neural network architecture that is particularly well suited for tasks that involve sequential data such as natural language processing. It uses self-attention mechanisms to allow the model to selectively focus on different parts of the input data, which allows it to learn to handle longer sequences of data and make more accurate predictions.
