Questions to explore

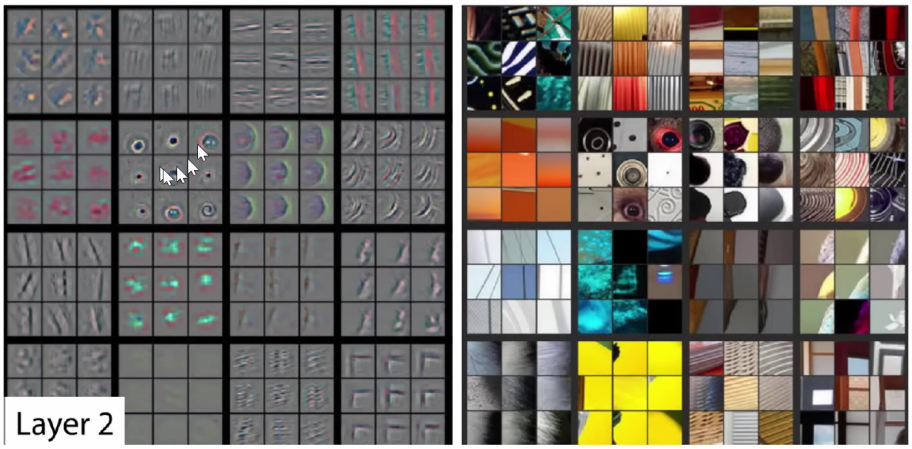
1. Difference between having multiple neurons in a single layer Vs distributing them across multiple layers.

We have seen in the first video of this course, under the topic '**visualizing layers of a trained neural network'** where it was explained how each layer is learning different features of the data pooling the learnt information to the next layer so that the next layer learns some more advanced features of the data.

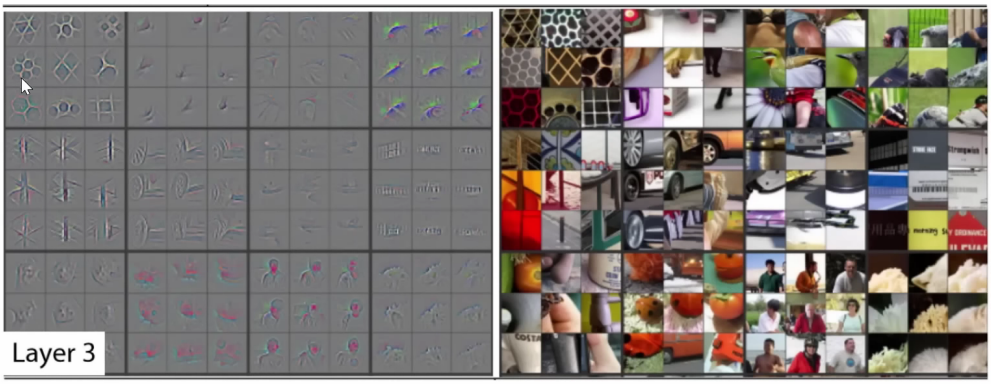
Layer one learning the edges and color gradients.

Images from Matthew D. Zeiler and Rob Fergus

Layer 2 learning corners and curves



Layer 3 learning some geometrical shapes



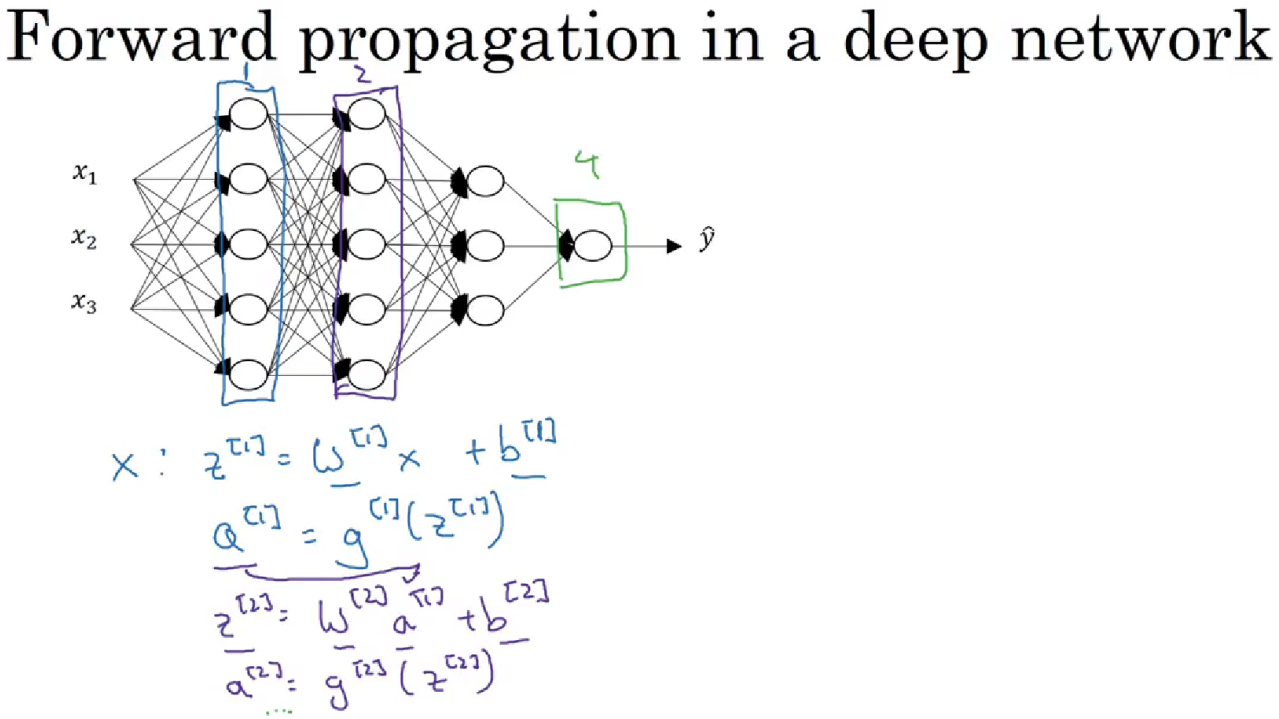
And so on.

If we use a single hidden layer in the neural network we might learn only the edges of the data (in case of images) and they aren't sent somewhere else to pool all the information for advanced features. The information learnt is projected out together. I believe that with this also we might get results, but for results equivalent to that of a deep network we need a lot of training, maybe an exponentially large number of times.

Mathematically

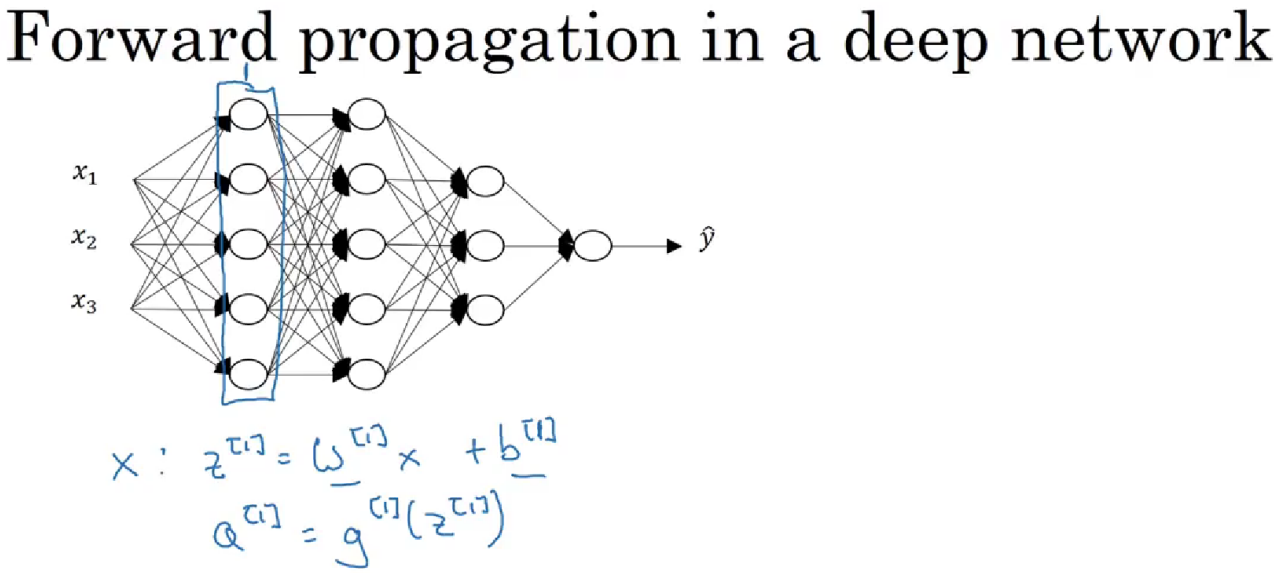
Note: Images below are not truly representing the question, but the written explanation and the equations in the images together will make sense for our context.

To see it slightly mathematically, let us say there are 2 hidden layers in our neural network and one layer produces an activation result to be sent to the next hidden layer



The next hidden layer is using the information shared or given by the previous hidden layer in the network to compute its values. Making the model understand more complex things rather than what a simple linear/logistic regression model might learn.

If we have a single layer with the number of neurons equal to the number of neurons in multiple layers we still be able to only learn as much as a simple linear/logistic regression model might learn.



We can see that the first layer is learning a simple model.

This is how we can say that a deep neural network with ‘n’ number of neurons distributed among layers of the network works better than a neural network with ‘n’ number of neurons in a single layer.

2. Generative Vs Discriminative

Generative models

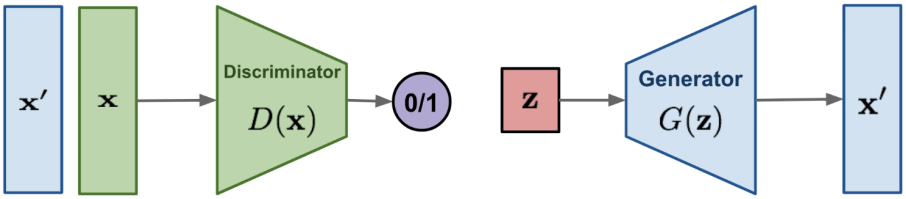
Generative models can generate new data instances, like new photos of animals that look like real animals. GANs are one kind of generative model.

Given a set of data instances X and a set of labels Y, Generative models capture the joint probability p(X, Y), or just p(X) if there are no labels.

A generative model includes the distribution of the data itself, and tells you how likely a given example is.

For example, models that predict the next word in a sequence are typically generative models, because they can assign a probability to a sequence of words.

GAN: minimax the classification error loss.



We can see that there are 2, x' and x given to the discriminator to check whether it is real or fake (generated). The generator produces from the learnt distribution. The work of the generator is to make it difficult for the discriminator to see which one is fake or real and go on improving its efficiency and discriminator tries to classify fake or real. Like this, both go hand in hand and improve the model.

Discriminative models

Discriminative models discriminate between different kinds of data instances. Discriminative model could tell a dog from a cat.

Given a set of data instances X and a set of labels Y, Discriminative models capture the conditional probability p(Y | X).

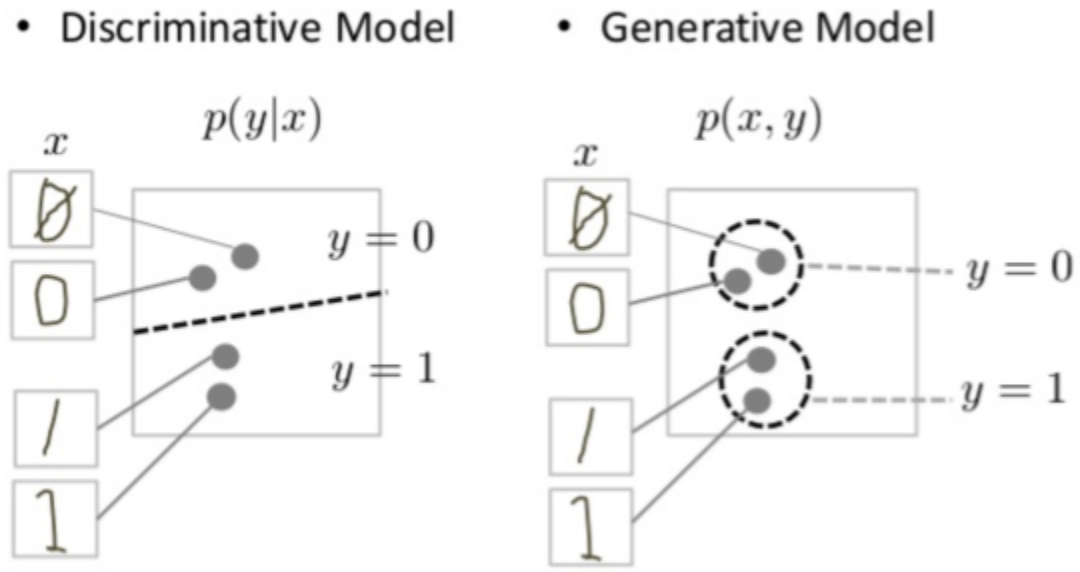
A discriminative model ignores the question of whether a given instance is likely, and just tells you how likely a label is to apply to the instance.

Differences

A generative model for images might capture correlations like "things that look like boats are probably going to appear near things that look like water" and "eyes are unlikely to appear on foreheads." These are very complicated distributions.

In contrast, a discriminative model might learn the difference between "sailboat" or "not sailboat" by just looking for a few tell-tale patterns. It could ignore many of the correlations that the generative model must get right.

Discriminative models try to draw boundaries in the data space, while generative models try to model how data is placed throughout the space.



3. Cross Entropy Loss, KL Divergence, Relation between KL Divergence and Entropy

Entropy

Entropy of a random variable X is the level of uncertainty inherent in the variable's possible outcome. For p(x) — probability distribution and a random variable X, entropy is defined as follows:

$$H(X) = \begin{cases} -\int_x p(x) \log p(x), & \text{if } X \text{ is continous} \\ -\sum_x p(x) \log p(x), & \text{if } X \text{ is discrete} \end{cases}$$

Cross-Entropy Loss Function

Also called logarithmic loss, log loss or logistic loss. Each predicted class probability is compared to the actual class desired output 0 or 1 and a score/loss is calculated that penalizes the probability based on how far it is from the actual expected value.

Cross-entropy loss is used when adjusting model weights during training. The aim is to minimize the loss for a better model. Cross-entropy is defined as

$$L_{\text{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \text{ for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

Binary Cross-Entropy Loss

It is a special case of cross entropy loss where there are only two classes. For binary classification (a classification task with two classes — 0 and 1), we have binary cross-entropy defined as

$$
\begin{aligned}
L &= -\sum_{i=1}^{2} t_i \log(p_i) \\
&= -[t_1 \log(p_1) + t_2 \log(p_2)] \\
&= -[t \log(p) + (1-t) \log(1-p)]
\end{aligned}
$$

where t_i is the truth value taking a value 0 or 1 and p_i is the Softmax probability for the i^{th} class. Since we have two classes 1 and 0 we can have $t_1 = 1$ and $t_2 = 0$ and since $p's$ are probabilities then $p_1 + p_2 = 1 \implies p_1 = 1 - p_2$. For the convenience of notation, we can then let $t_1 = t, t_2 = 1 - t, p_1 = p$ and $p_2 = 1 - p$.

KL Divergence

The Kullback–Leibler divergence (also called relative entropy and I-divergence), denoted DKL(P // Q), is a type of statistical distance: a measure of how one probability distribution P is different from a second, reference probability distribution Q.

For discrete probability distributions P and Q defined on the same sample space, $\mathcal{X}$, the relative entropy from Q to P is defined[11] to be

$$
D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right),
$$

which is equivalent to

$$
D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right).
$$

For distributions P and Q of a continuous random variable, relative entropy is defined to be the integral[14]

$$
D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx,
$$

where p and q denote the probability densities of P and Q.

Relation Between KL divergence and Entropy

KL divergence is the relative entropy or difference between cross entropy and entropy or some distance between actual probability distribution and predicted probability distribution.

References:

- Coursera, Deep Learning Specialization by Andrew Ng from Deeplearning.ai, first course, week 4, video 2.
- Deep Learning Part - II (CS7015) Lec 22.1 Generative Adversarial Networks - The Intuition
- https://developers.google.com/machine-learning/gan/generative#:~:text=A%20generative%20model%20includes%20the,to%20a%20sequence%20of%20words.
- https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e
- https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
- https://towardsdatascience.com/entropy-cross-entropy-and-kl-divergence-17138ffab87b#:~:text=KL%20divergence%20is%20the%20relative,as%20the%20actual%20probability%20distribution.