Name: Utkarsh ①
Roll no: M20 AI5318

Q1) RNN (LSTM)

1. Weights can be shared across time steps
2. can process inputs on any length
3. model size won't increase if input size is large
4. LSTM (long term short memory) will try to remember information for longer periods of time.
5. particularly useful for speech recognitions, time series tasks.

Q5) The answer for this cannot be simply characterized to a single option because both the approaches have their own merits. Optimization with ~~helps the~~ for pre training will help in better parameter initization which overall lead to better fine tuning of network. This will hence lead to choosing of optimal weights. Regularization on the other end will show that searching would be more structured.

Q3) Stochastic Gradient Descent (SGD)

works with the process of chance or probability randomly where random samples in small size are selected rather than whole data set in each iteration. It is better than Gradient Descent because it uses only a single sample. Here we try to find cost function of a single example at each iteration. As we choose only one sample the path to minima will be not smooth but as our target is to for just find minima if we reach in shortest time we are good.

Because it requires frequent "serial training time" and "scanning the whole training set in many passes" to reach the desired asymptotic region, the process would be difficult to scale for large data sets.

The training time overall can be reduced by either

i) reducing epochs of training
ii) exploring some new algo which can perform the same task in distributed manner.

~~Some other~~
Instead of SGD we can opt for "Average SGD" which ~~will~~ can significantly reduce the training time due to it's one line and one pass learning.
There are other algo's like Asynchronous SGD, Hessian-free optimization that can help us achieve this task.

Q9)

Yes, we can use NAS or Neural architecture search for model compression of pre-trained models. As the idea is always to reach a goal where human intervention should be removed from the whole process, this being a symmetric and automized approach to learn the model's optimal architecture helps us achieve that. NAS in simple sense is just a searching technique over various neural network components.

Q10)

Approach from image

1. Feature extractor (green)
2. label predictor (blue)
3. domain classifier (red)
4. This is a feed forward architecture due to feature extractor and label predictor working together while domain classifier makes it a unsupervised domain adaptation. This is because domain classifier and feature extractor connect using a gradient reversal layer which multiplies the gradient by a certain negative constant while back propagation is performed. Gradient reversal also ensures that features which are distributed over the given two domains are ~~sim~~ as similar as possible which leads to domain-invariant features.

Q11)

To prove,

$$\log_e(p(x)) \geq \mathbb{E}_{z \sim q(z)}[\log p(x/z))] - D_{KL}(q(z) || p(z))$$

from KL Divergence we know,

$$D_{KL}(q_\phi(z/x) || P(z/x)) = \int q_\phi(z/x) \log\left(\frac{q_\phi(z/x)}{p(z/x)}\right) dz$$

$$= \int q_\phi(z|x) \log \frac{q_\phi(z|x)\, p(x)}{p(x,z)}\, dz$$

$$= \int q_\phi(z|x) \left(\log p(x) + \log \frac{q_\phi(z|x)}{p(x,z)} \right) dz$$

$$= \int q_\phi(z|x) \log p(x)\, dz + \int q_\phi(z|x) \log \frac{(q_\phi(z|x))}{p(x|z)\, p(z)}\, dz$$

$$= \log p(x) + \int q_\phi(z|x) \log \frac{q_\phi(z|x)}{p(z)} - \int q_\phi(z|x) \log p(x|z)\, dz$$

$$= \log p(x) + D_{KL}(q_\phi(z|x) \,||\, p(z)) - E_{z \sim q_\phi(z|x)}[\log p(x|z)]$$

Q 7) · Incremental Training

① Designed to solve problem where NN is exposed to a changing environment. where new incoming o/p attribute are introduced.

② Goal is to
   a) improve accuracy
   b) reduce network complexity

③ Network needs to grow it's capacity with arrival of data of new classes.

④ Training methods idea is to reduce the interference within the given inputs which increases performance.

⑤ E

Q12)

Given,

$$l_{GAN}(F, D_x) = E_x[\log D_x(x)] + E_y[\log(1 - D_x(F(y)))]$$

$$l_{GAN}(G, D_y) = E_y[\log D_y(y)] + E_x[\log(1 - D_y(G(x)))]$$

Mapping $x \to y$

To find, optimal discriminator D_x^* & maxing $l_{GAN}(F, D_x)$

$$\min_F \max_{D_x} \left(E_{x \sim p\,data(y)} (\log D_y(y)) + E_x \sim p\,data(y) (\log(1 - D_x F(y))) \right)$$

$$\min_F \max_{D_x} \left[E_x - p(data(x)) (\log D_x(x)) + E_y \sim \cancel{p(data)}\ p\,data(y) \log(1 - D_x(F_y)) \right.$$

$$\Rightarrow \min_F \max_{D_x} \int_x P_{data}(x) \log D_x(x) + P_f(x) \log(1 - D_x(x))]\, dx$$

$$\Rightarrow \min_F \underbrace{\int \max_{D_x} (P\,data(x) \log D_x(x)) + P_f(x) \log(1 - D_x(x))]\, dx}$$

from here

The optimal discriminator $D_x^*(x) = \cancel{P\,data}\ \dfrac{P_{data}(x)}{P_{data}(x) + P_f(x)}$