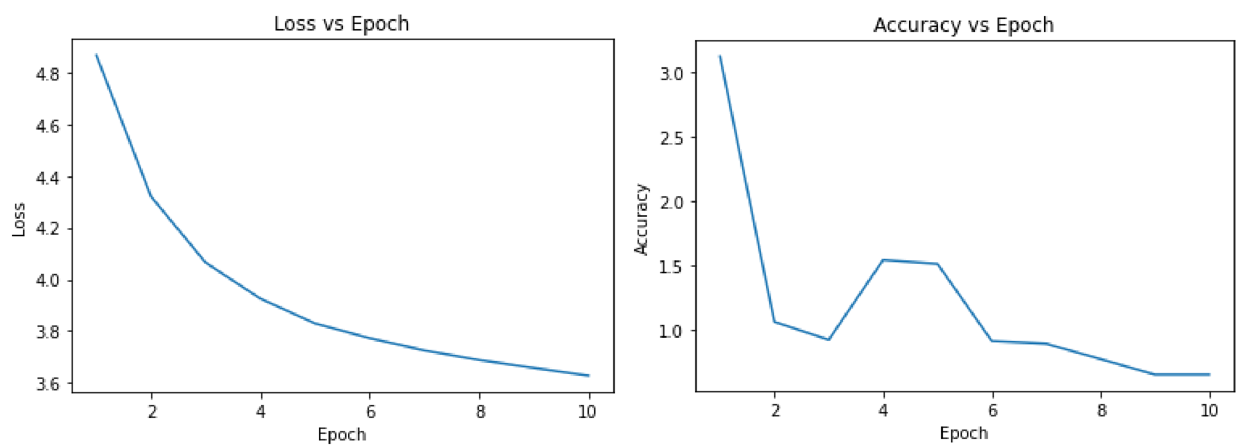
DEEP LEARNING MINOR-2 REPORT

Google Colab:

https://colab.research.google.com/drive/1IDBVumDREO_0WXJOHYetXxUSOdius7Xn?usp=sharing

QUESTION-1:

These are the results obtained from the teacher model:



Performance of teacher model:

1)High Loss: teacher model loss is very high

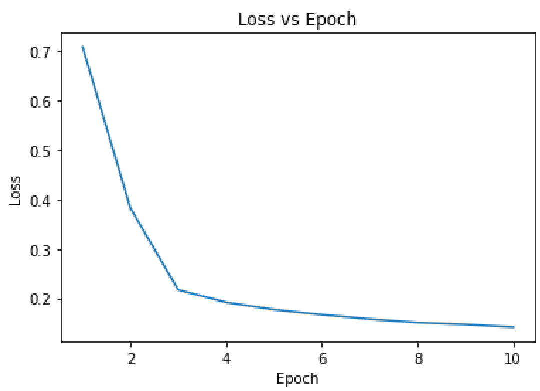
2)Loss is decreasing very slowly with each epoch

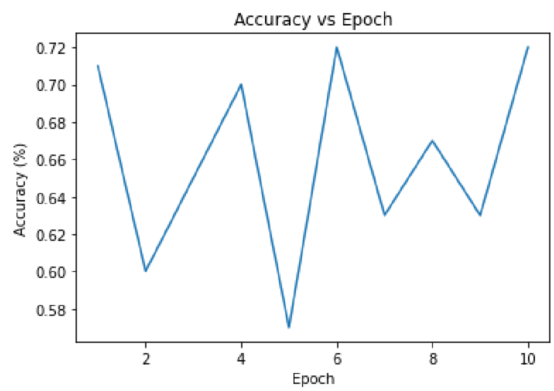
3)Accuracy on the validation set is very less indicating the model is not able to classify properly

4) Highest loss:4.966 (epoch 1)

Lowest loss: 3.627(epoch 10)

These are the results obtained from the student model:





Highest Loss - Epoch 1 Loss: 0.7080
Lowest Loss - Epoch 10 Loss: 0.1420

Performance:

1) Loss is decreasing over every epoch. This indicates that model is learning but it is learning slowly.
2) 2)The accuracy on the validation set fluctuates between 0.57% to 0.72% over the training epochs. This is very less which indicates that performance is bad.

Observations:

1) Loss decreases gradually after every epoch, but accuracy on the validation set is not improving. This is caused by overfitting
2) Reason for overfitting is a simple architecture with less complexity and a number of classes which is 200 (it is more than the batch size)

Results obtained using the exponential moving average method:

Epoch: 1, Train Loss: 0.254, Validation Loss: 5.210547

Epoch: 2, Train Loss: 0.361, Validation Loss: 5.210424

Epoch: 3, Train Loss: 0.199, Validation Loss: 5.210392

Epoch: 4, Train Loss: 0.12, Validation Loss: 5.210539

Epoch: 5, Train Loss: 0.144, Validation Loss: 5.210430

From this results, we can say that the student model without using the exponential average method performs better than this.

Comparison between student and teacher model:

Loss: The teacher model has a much higher loss than the student model, indicating that the teacher model is not performing as well as the student model, because the teacher model is more complex and requires more training to converge.

Accuracy: The accuracy of the teacher model on the validation set is low, whereas the accuracy of the student model fluctuates but is overall higher than that of the teacher model. This suggests that the student model is able to generalize better than the teacher model.

Training time: student model requires less time than the teacher model.

Overfitting: Both models overfit to some extent, as the accuracy on the validation set is lower than the accuracy on the training set. But, the gap between the two accuracies is smaller for the student model, which implies that it is less prone to overfitting than the teacher model.

CONCLUSION: The student model is outperforming the teacher model in this case. so, prefer the student model

QUESTION-2

QUESTION-A:

How can a convolutional neural network be effectively designed and trained to automatically learn and extract high-level features from complex input data, robustly generalise to new, unseen data, avoid overfitting, and minimize computational complexity and efficiency?

The answer of ChatGPT :

Several approaches can be used to effectively design and train a convolutional neural network (CNN) that can automatically learn and extract high-level features from complex input data while also being able to robustly generalise to new, unseen data and avoid overfitting while minimising computational complexity and maximising computational efficiency.

Architecture Design: The architecture of the CNN plays a critical role in its ability to learn complex features and generalize well to new data. One effective approach is to use a deep architecture with multiple convolutional and pooling layers, which can capture both low-level and high-level features. Additionally, using skip connections, residual blocks, or dense connections can improve the flow of information through the network and reduce vanishing gradients. Incorporating attention mechanisms or transformers can help the network focus on the most relevant features for a given task.

Regularization Techniques: To avoid overfitting, regularization techniques can be used during training. One common approach is to use dropout, which randomly drops out some neurons during training to prevent co-adaptation. L1/L2 regularization can also be used to add a penalty term to the loss function, which encourages the network to learn simpler representations. Early stopping, where training is stopped when validation error stops improving, can also help prevent overfitting.

Data Augmentation: Data augmentation techniques can be used to increase the diversity of the training set, which can improve the generalization of the model. Techniques such as cropping, flipping, and rotating can be used to create new variations of the input data. Additionally, techniques such as Mixup and CutMix can be used to blend multiple samples together, creating a more diverse set of training examples.

Transfer Learning: Transfer learning can be used to leverage pre-trained models to improve the performance of a CNN on a new task. By using a pre-trained model as a feature extractor and fine-tuning the last few layers for the specific task, the network can be trained with a smaller dataset and achieve higher accuracy.

Optimization Algorithms: Optimization algorithms such as stochastic gradient descent (SGD) with momentum, Adam, or RMSProp can be used to optimize the weights of the network during training. Using a learning rate schedule or adaptive learning rate can help the network converge faster and more reliably.

Overall, by carefully designing the architecture, using regularization and data augmentation techniques, leveraging pre-trained models through transfer learning, and using effective optimization algorithms, it is possible to design and train a CNN that can automatically learn and extract high-level features from complex input data, generalize well to new, unseen data, avoid overfitting, and maximize computational efficiency.

Justification:

Overfitting is a common problem in deep learning models, including CNNs, where the model learns the training data too well and cannot generalize to new data. Regularization techniques are used to limit the complexity of the model and prevent overfitting. These techniques help improve the generalization ability of the model. For example, dropout is a commonly used regularization technique in which some neurons are randomly dropped out during training, forcing the model to learn more robust and independent features. L1/L2 regularization, on the other hand, adds a penalty term to the loss function, which encourages the model to learn simpler weights. Another popular procedure is early stopping, which interrupts the training process when the validation loss begins to increase, preventing the model from overfitting.

importance of optimization algorithms in training CNNs, consider an image classification. Suppose we have a dataset of 10,000 images of various objects and want to train a CNN to classify these objects into 10 different categories. To train the CNN, we need to initialize the weights of the network randomly and then adjust them during training to minimize the loss function. The loss function measures how well the CNN predicts the correct category for each image. The goal of the optimization algorithm is to adjust the weights to minimize the loss function. If we use a simple optimization algorithm such as stochastic gradient descent (SGD), it may take a very long time for the CNN to converge to the optimal solution, and there is a risk that it will get stuck in a local optimum. This means that CNN may not be able to learn the most important features for accurately classifying the objects in the dataset. However, if we use a more advanced optimization algorithm such as Adam, Adagrad, or RMSProp, the CNN can converge to the optimal solution much faster and avoid getting stuck in local optima. For example, Adam is an adaptive learning rate optimization algorithm that adjusts the learning rate for each weight based on its history, which helps to achieve faster convergence and avoid local optima.

QUESTION-B:

can student-teacher learning be used to improve the accuracy of chatgpt? how will it improve?

Answer: Yes, a student-teacher model could be used to improve the accuracy of ChatGPT.

ChatGPT is a large language model trained on a massive amount of data, which has enabled it to achieve impressive results in a wide range of natural language processing (NLP) tasks, including question-answering, text summarization, and language translation. However, even with its large size and extensive training, ChatGPT may still face challenges in some specific NLP tasks, such as generating responses that are relevant and informative in a conversational setting.

To improve the accuracy of ChatGPT in these specific tasks, a student-teacher model could be used to transfer the knowledge and expertise from a more specialized teacher model to the ChatGPT student model. For example, a teacher model could be trained on a large dataset of conversational data, and the student model could be trained to mimic the behavior of the teacher model in generating high-quality responses.

The teacher model could be a simpler, more specialized model that is designed specifically for conversational tasks, while the ChatGPT student model could be a larger and more complex model that has been pre-trained on a wide range of NLP tasks. By using the teacher model as a guide, the ChatGPT student model could learn to generate responses that are more relevant and informative in a conversational setting.

During training, the student model would be trained to mimic the behavior of the teacher model, which could be done using a knowledge distillation loss function. This loss function measures the difference between the outputs of the teacher and student models, and the goal is to train the student model to produce similar outputs to the teacher model.

In summary, a student-teacher model could be used to improve the accuracy of ChatGPT in specific NLP tasks, such as generating relevant and informative responses in a conversational setting. By using a teacher model trained on specialized conversational data, the ChatGPT student model could learn to generate responses that are more relevant and informative, while still leveraging the knowledge and expertise gained from pre-training on a wide range of NLP tasks.

QUESTION-C:

You are given a multiclass classification problem. In neural networks, The number of output classes is 4.

True label = rabbit

Input values for the SoftMax function : truck = 1, bird=4, rabbit =8, car=2

Find the cross-entropy loss and use this loss to calculate the result of the first iteration of the exponential average method

assuming hyperparameter alpha=0.4. Initialise loss0=0 and find the result obtained by the exponential average method at the end of the first iteration.

ChatGPT solution: **this solution is wrong**

To calculate the cross-entropy loss for the given example,

we use the formula: $H(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y_i})$ where y is the true label and $\hat{y}$ is the predicted label. For the given example, the true label is "rabbit" and the predicted label has highest probability for "rabbit". Therefore, $y=[0, 0, 1, 0]$ and $\hat{y}=[0, 0, 1, 0]$. So the cross-entropy loss is: $H(y, \hat{y}) = - (0 \times \log(0) + 0 \times \log(0) + 1 \times \log(1) + 0 \times \log(0)) = 0$ Now, using the exponential average method with alpha=0.4, the result at the end of the first iteration can be calculated as: $loss_{avg}^{(1)} = \alpha \times loss^{(1)} + (1-\alpha) \times loss_{avg}^{(0)}$ where $loss^{(1)}$ is the loss from the first iteration and $loss_{avg}^{(0)}$ is the initial value of the loss average, which is set to 0. Substituting the values, we get: $loss_{avg}^{(1)} = 0.4

\times 0 + 0.6 \times 0 = 0$ Therefore, the result obtained by the exponential average method at the end of the first iteration is 0

Correct solution:

Cross entropy loss = -t1(ln(SoftMax(car)))- t2(ln(SoftMax(truck))- t3(ln(SoftMax(bird)))- t4(ln(SoftMax(rabbit)))

SoftMax(rabbit)=(e^8) /$(e^2+e+e^4+e^8)$ =0.978

T1=t2=t3=0 since they are not true labels

T4=1

Cross entropy loss = -t4(ln (0.978)) =-ln (0.978) = 0.0214

Exponential average method(EAM)= alpha*loss1 +(1-alpha)*loss0

Loss1 = 0.0214

Loss0 = 0 (given)

EAM = 0.4*0.0214+(1-0.4) *0 = 0.00856