Preface:

Models are too large to upload to black board and thus kept local. By running the CNN_GPU python notebook, it can download the required data. The inceptionV4 notebook can download the label used for imagenet, and it need the CNN_GPU to download the tiny imagenet for the data. I'll try to include the tiny imagenet dataset but I don't know if blackboard is going to like it.

CNN

For inception net, I'm using V3 instead of V4. (V4 is only used for not finetuned.) V3 is more mature and has better support on tensorFlow, will V4 is only available with timm package on hugging face. Thus I also included V3 for comparison

All models: VGG16, VGG19, Resnet50, Resnet100 and InceptionV3 are included from the tensorFlow models. The version of tensorFlow I'm using is 2.10.0 because it stopped supporting GPU after that version on Windows Native. All three networks are appended with a flatten then dense maxpool layer with softmax activation function, and 200 nodes as output. All but the output layers are hold to be untrainable, loaded with pre-trained weights ("DEFAULT" is trained on imageNet), then finetuned to work with tiny imagenet dataset. The accuracy is determined by the top 1 category (instead of "within the top 5"). Which explains why the accuracy is generally not high.

Before I figured out GPU setup with tensorflow, I used CPU to train 10epochs for VGG19, ResNET50 and InceptionV3. The finetuning training time is reported in the table below.

Model	VGG16	VGG19	ResNET50	ResNET100	InceptionV3
Trainable Parameters Count	5,017,800	5,017,800	409,800	409,800	409,800
Toal Parameters Count	19,732,488	25,042,184	23,974,600	43,036,360	22,212,584
GPU train time (1080ti, per epoch)	440s	435s	430s	435s	775s
CPU train time (AMD Ryzen 9 '7900X, per epoch)	N/A	1150s	1570s	N/A	835s

Table 1 - Models and their basic stats

Models		VGG16	VGG19	Resnet50	Resnet100	InceptionV3
15 epochs	**Accuracy**	30.07%	30.14%	5.34%	6.85%	3.08%
	Loss	119.83	112.19	76.73	19.48	31.08
50 epochs	**Accuracy**	31.35%	30.45%	11.9%	9.66%	5.58%
	Loss	81.91	75.91	12.00	8.8045	6.73
70 epochs	**Accuracy**	31.2%	30.54%	14.31%	9.74%	8.55%
	Loss	71.52	66.79	8.05	8.79	4.97

Table 2 - Finetuning results

Models	VGG16	VGG19	Resnet50	Resnet100	InceptionV3	InceptionV4
Accuracy	24.5%	25.3%	33.3%	42.04%	49.8%	46.84%

Table 3 - Direct Application Accuracy Result (No Finetuning)

It seems like larger models such as resnet50, resnet100, inception net needs more epochs to finetune the model and provide a good result: For the same number of epochs trained, VGG19 has significantly better accuracy than resnet50, and same relationship holds between VGG16 and resnet100 or inceptionV3. Between the same family of models, VGG16 reached similar accuracy only after 15 epochs, comparing VGG19's 50. For resnet family, resnet100 after 15epochs has significantly lower accuracy than resnet50. For inceptionV3, after 50 epochs, it reaches similar accuracy compared to that of Resnet50.

The reason behind this performance difference between families of model is likely due to how simple that family of models are. VGG16 and 19 are much simpler than resnets or inception nets. While VGG is only convolution layers followed by pooling layers, resnet use residual blocks for deeper network, inception net even more complex. The simplicity of the model makes the model easier to achieve better results with limited fine-tuning training time.

Within each family, the smaller the network, the easier it is to fine tune and better the result is with limited epochs. VGG16 only needed 15 epochs to reach the similar accuracy of VGG19. Looking at resnet50's 15th epoch (accuracy 0.1136, loss 13.3814), it's also way ahead of resnet100's 15th epoch. VGG family was able to reach a stable accuracy score within the 70 epochs, while Resnet and inceptionV3 still shows increase in growth after the total epochs is reached. This is due to the smaller size of the network, makes it less complex and thus easier to finetune.

For inceptionV4, after looking at the low accuracy result from inceptionV3 (and the fact I don't have nearly enough time and resource to finetune it), I directly checked it on the validation set. Reaching an accuracy of 0.498, the highest amongst the bunch, it seems like sometimes unfinished fine-tuning will yield worse results than just applying the model to the dataset. This is because when we finetune the model from 1000 classes to 200 classes, we are required to train a dense layer at the output, which consists 5,017,800 nodes for VGG or 409,800 nodes for ResNet/Inception to train. From the trend of Resnet and InceptionNet's accuracy, it seems that if I have enough resources, finetuning will eventually yields a better accuracy result than no finetuning.

References:

[1]Simonyan, K. and Zisserman, A., “Very Deep Convolutional Networks for Large-Scale Image Recognition”, 2014. doi:10.48550/arXiv.1409.1556.

[2] C. Szegedy, et al., "Going deeper with convolutions," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015 pp. 1-9. doi: 10.1109/CVPR.2015.7298594

[3]Shafiq M, Gu Z. Deep Residual Learning for Image Recognition: A Survey. Applied Sciences. 2022; 12(18):8972. https://doi.org/10.3390/app12188972