

### Question1:

**Dataset Used:** STL-10

**Procedure:** Extracted features from last FC layer of Resnet50. Use these extracted feature embeddings to train Multiclass SVM classifier as One-Vs-Rest classifier.

**Observations:**

(a)

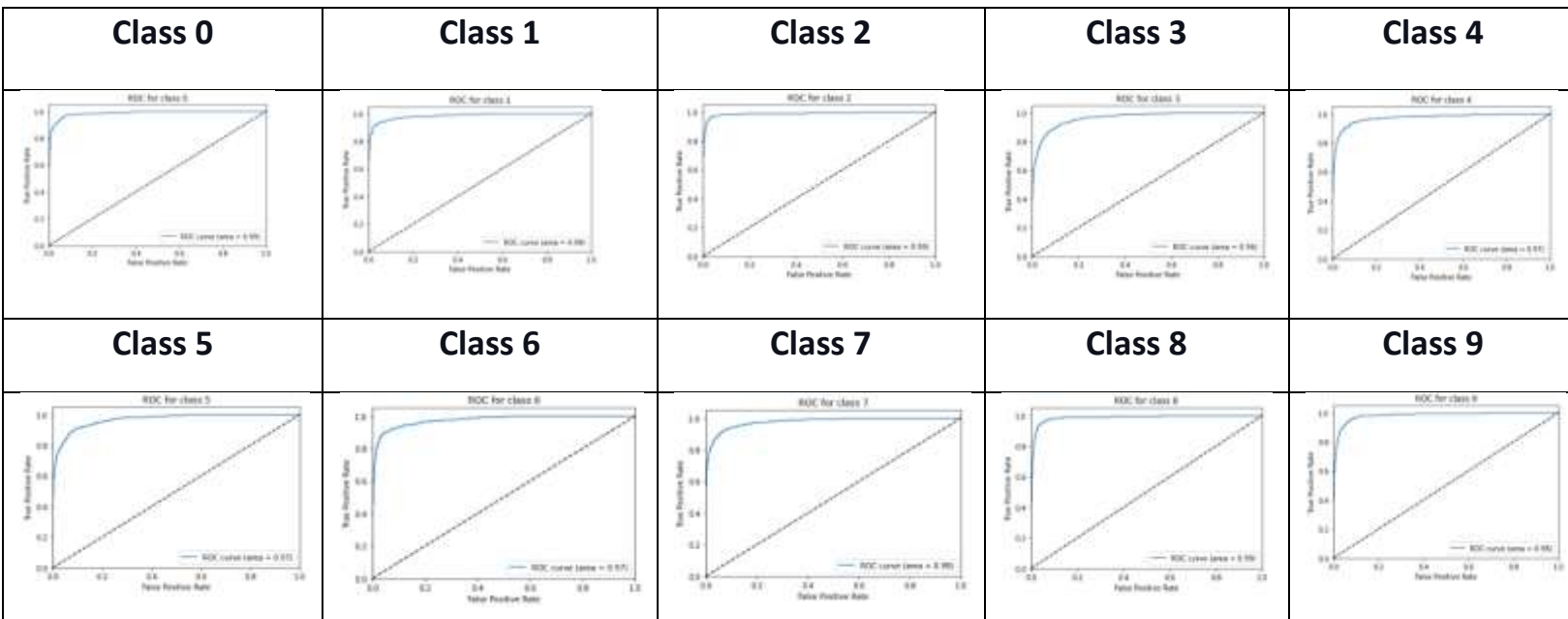
1. Test Accuracy: 83.65 %
2. Confusion Matrix:

Actual

Predicted

	0	1	2	3	4	5	6	7	8	9
0	679	12	12	0	4	1	2	2	69	19
1	10	709	1	24	12	6	3	32	2	1
2	9	0	719	0	3	1	1	1	3	63
3	3	15	3	586	60	62	11	49	5	6
4	4	21	0	34	663	20	42	13	2	1
5	5	20	2	38	23	624	62	21	4	1
6	8	3	5	7	48	36	669	14	3	7
7	1	23	3	34	41	14	15	664	4	1
8	38	1	3	2	0	0	3	0	710	43
9	20	0	39	3	2	1	7	7	52	669

### 3. ROC Curve:



**(b) I. Before Fine-tuning Resnet50**

**Procedure:** Predicted the STL-10 classes from Resnet50 trained on Imagenet, by taking the STL-10 classes in use and kept other classes which are in Imagenet and not in STL-10 as ***'other'*** class name. So there will be 11 classes ['airplane', 'bird', 'car', 'cat', 'deer', 'dog', 'horse', 'monkey', 'ship', 'truck', 'other']

1. **Test Accuracy:** 20.75 %
2. **Confusion Matrix:**

[illegible]

### 3. Class-wise Accuracy:

Class	Airplane (0)	Bird (1)	Car (2)	Cat (3)	Deer (4)	Dog (5)	Horse (6)	Monkey (7)	Ship (8)	Truck (9)
Accuracy	96.89%	94.28%	96.72%	95.56%	0.0%	73.07%	96.80%	97.73%	80.34%	86.05%

## II. After Fine-tuning Resnet50

**Procedure:** Fine-tuned Resnet50 with STL-10 classes by freezing some of the layers of Resnet50 and changing last FC layer neurons to 50 neurons.

1. **Test Accuracy:** 91.55 %

2. **Confusion Matrix:**

		Actual									
Predicted		0	1	2	3	4	5	6	7	8	9
	0	737	5	5	0	0	0	0	1	37	15
	1	2	755	0	18	3	4	2	16	0	0
	2	4	0	750	0	0	0	1	0	2	43
	3	1	6	0	682	32	41	3	34	1	0
	4	0	12	0	22	735	11	17	3	0	0
	5	1	3	0	45	16	673	33	28	1	0
	6	1	3	0	4	33	37	710	7	1	4
	7	0	9	0	10	13	9	3	755	1	0
	8	9	0	0	0	0	0	0	0	769	22
	9	5	0	17	0	0	0	0	1	19	758

### 3. Class-wise Accuracy:

Class	Airplane (0)	Bird (1)	Car (2)	Cat (3)	Deer (4)	Dog (5)	Horse (6)	Monkey (7)	Ship (8)	Truck (9)
Accuracy	96.97%	95.20%	97.15%	87.32%	88.34%	86.83%	92.32%	89.34%	92.53%	90.02%

## Conclusion:

- On the basis of Comparison of above models we can say that Fine-tuned Resnet50 performs better than before fine-tuned resnet50 and trained SVM.

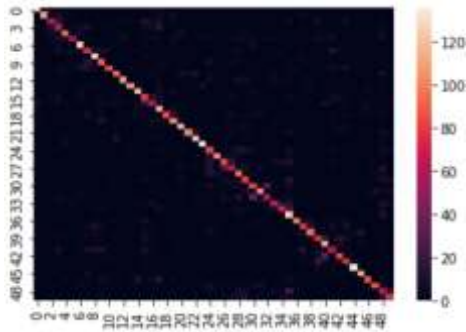
## **Question2:**

**Dataset Used:** Tiny Imagenet

**Procedure:** Used first 50 classes of Tiny imagenet out of 200 classes to Fine-tune Resnet18 with various loss functions.

### **I. Fine-tuning using Triplet loss:**

1. **Loss** = Categorical Cross-Entropy loss + **1.0 x** Triplet loss
2. **Learning rate:** 0.0001
3. **Epochs:** 11
4. **Test Accuracy:** 61.67 %
5. **Confusion Matrix:**

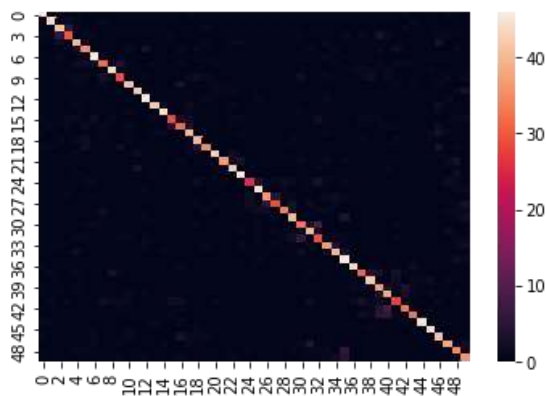


### **6. Classification Report:**

Classification Report				
	precision	recall	f1-score	support
0	0.84	0.87	0.85	150
1	0.92	0.75	0.82	150
2	0.65	0.43	0.52	150
3	0.41	0.43	0.42	150
4	0.67	0.86	0.60	150
5	0.69	0.37	0.48	150
6	0.68	0.85	0.75	150
7	0.47	0.50	0.49	150
8	0.69	0.85	0.76	150
9	0.49	0.59	0.54	150
10	0.72	0.55	0.61	150
11	0.66	0.65	0.66	150
12	0.92	0.72	0.81	150
13	0.81	0.73	0.77	150
14	0.72	0.79	0.75	150
15	0.49	0.55	0.53	150
16	0.44	0.43	0.43	150
17	0.51	0.79	0.62	150
18	0.82	0.56	0.67	150
19	0.64	0.68	0.66	150
20	0.05	0.79	0.12	150
21	0.51	0.68	0.58	150
22	0.71	0.81	0.76	150
23	0.87	0.91	0.89	150
24	0.44	0.55	0.49	150
25	0.84	0.71	0.77	150
26	0.61	0.56	0.59	150
27	0.45	0.43	0.44	150
28	0.48	0.65	0.55	150
29	0.47	0.55	0.51	150
30	0.51	0.56	0.54	150
31	0.66	0.69	0.68	150
32	0.54	0.38	0.45	150
33	0.59	0.46	0.52	150
34	0.72	0.50	0.59	150
35	0.47	0.83	0.60	150
36	0.92	0.68	0.78	150
37	0.59	0.54	0.56	150
38	0.71	0.63	0.67	150
39	0.56	0.32	0.42	150
40	0.54	0.76	0.63	150
41	0.46	0.41	0.44	150
42	0.57	0.50	0.53	150
43	0.54	0.53	0.54	150
44	0.92	0.91	0.91	150
45	0.81	0.77	0.79	150
46	0.74	0.58	0.65	150
47	0.81	0.59	0.68	150
48	0.32	0.46	0.38	150
49	0.58	0.51	0.55	150
accuracy			0.63	7500
macro avg	0.64	0.63	0.63	7500
weighted avg	0.64	0.63	0.63	7500

## II. Fine-tuning using Categorical-Cross Entropy loss:

1. Loss = Categorical Cross-Entropy loss
2. Learning rate: 0.00001
3. Epochs: 12
4. Test Accuracy: 73.67 %
5. Confusion Matrix:

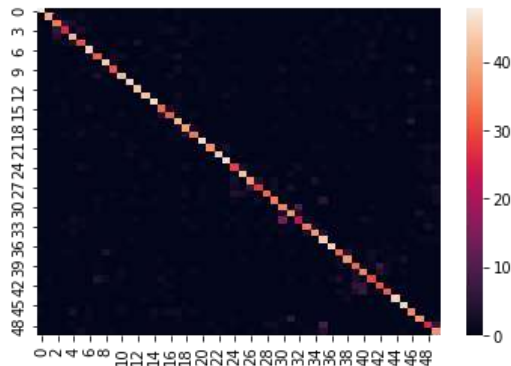


## 6. Classification Report:

Classification Report				
	precision	recall	f1-score	support
0	0.87	0.92	0.89	50
1	0.88	0.88	0.88	50
2	0.76	0.82	0.79	50
3	0.70	0.60	0.65	50
4	0.67	0.80	0.73	50
5	0.70	0.70	0.70	50
6	0.82	0.92	0.87	50
7	0.71	0.64	0.67	50
8	0.88	0.88	0.88	50
9	0.69	0.58	0.63	50
10	0.80	0.82	0.81	50
11	0.89	0.84	0.87	50
12	0.81	0.92	0.86	50
13	0.89	0.84	0.87	50
14	0.90	0.90	0.90	50
15	0.76	0.58	0.66	50
16	0.64	0.64	0.64	50
17	0.71	0.80	0.75	50
18	0.75	0.80	0.78	50
19	0.83	0.70	0.76	50
20	0.88	0.86	0.87	50
21	0.95	0.74	0.83	50
22	0.82	0.84	0.83	50
23	0.84	0.92	0.88	50
24	0.70	0.52	0.63	50
25	0.79	0.90	0.84	50
26	0.78	0.72	0.75	50
27	0.76	0.58	0.66	50
28	0.87	0.68	0.76	50
29	0.75	0.80	0.78	50
30	0.67	0.66	0.67	50
31	0.71	0.80	0.75	50
32	0.60	0.58	0.59	50
33	0.73	0.74	0.73	50
34	0.77	0.80	0.78	50
35	0.67	0.92	0.77	50
36	0.90	0.88	0.89	50
37	0.74	0.62	0.67	50
38	0.75	0.84	0.79	50
39	0.59	0.74	0.65	50
40	0.63	0.76	0.69	50
41	0.65	0.56	0.60	50
42	0.72	0.66	0.69	50
43	0.79	0.66	0.72	50
44	0.98	0.90	0.94	50
45	0.88	0.90	0.89	50
46	0.69	0.82	0.75	50
47	0.76	0.76	0.76	50
48	0.60	0.70	0.65	50
49	0.64	0.72	0.68	50
accuracy			0.76	2500
macro avg	0.77	0.76	0.76	2500
weighted avg	0.77	0.76	0.76	2500

## III. Fine-tuning using Center loss:

1. **Loss** = Categorical Cross-Entropy loss + **0.1 x Center loss**
2. **Learning rate:** 0.00001
3. **Epochs:** 11
4. **Test Accuracy:** 72.61 %
5. **Confusion Matrix:**



## 6. Classification Report:

Classification Report				
	precision	recall	f1-score	support
0	0.89	0.96	0.92	50
1	0.90	0.88	0.88	50
2	0.70	0.70	0.70	50
3	0.65	0.52	0.58	50
4	0.66	0.82	0.73	50
5	0.65	0.60	0.63	50
6	0.79	0.90	0.84	50
7	0.70	0.64	0.67	50
8	0.85	0.88	0.86	50
9	0.74	0.62	0.67	50
10	0.81	0.64	0.72	50
11	0.88	0.90	0.89	50
12	0.84	0.86	0.85	50
13	0.90	0.86	0.88	50
14	0.85	0.92	0.88	50
15	0.69	0.66	0.67	50
16	0.68	0.64	0.66	50
17	0.67	0.84	0.74	50
18	0.74	0.78	0.76	50
19	0.77	0.68	0.72	50
20	0.82	0.92	0.87	50
21	0.97	0.74	0.84	50
22	0.94	0.88	0.91	50
23	0.81	0.94	0.87	50
24	0.63	0.58	0.60	50
25	0.80	0.86	0.83	50
26	0.79	0.76	0.78	50
27	0.78	0.56	0.65	50
28	0.87	0.60	0.72	50
29	0.74	0.70	0.72	50
30	0.59	0.72	0.65	50
31	0.70	0.76	0.73	50
32	0.56	0.44	0.49	50
33	0.78	0.70	0.74	50
34	0.76	0.76	0.76	50
35	0.64	0.88	0.74	50
36	0.90	0.90	0.90	50
37	0.70	0.66	0.68	50
38	0.72	0.76	0.74	50
39	0.57	0.66	0.61	50
40	0.62	0.72	0.67	50
41	0.64	0.68	0.62	50
42	0.65	0.62	0.63	50
43	0.73	0.64	0.68	50
44	0.96	0.98	0.93	50
45	0.84	0.92	0.88	50
46	0.65	0.72	0.69	50
47	0.83	0.76	0.79	50
48	0.57	0.52	0.54	50
49	0.63	0.74	0.68	50
accuracy			0.75	2500
macro avg	0.75	0.75	0.75	2500
weighted avg	0.75	0.75	0.75	2500

## Conclusion:

- On Comparing models on the basis of various metrics we can say that the model trained with Categorical Cross-Entropy loss has better performance for Test data as compared to model trained with Triplet loss and Center loss for classification task.
- But we can see some interesting part above that model trained with Center loss performs significantly similar to model trained with Categorical Cross-Entropy loss.

### **Question3:**

**Dataset Used:** Dogs Vs Cats

**Train-Test Split:** 70:30

#### **(a) i. Vanilla SGD**

1. **Batch size:** 1
2. **Test Accuracy:** 43.24 %
3. **Learning rate:** 0.001
4. **Epochs:** 5

#### **ii. Mini Batch SGD**

1. **Batch size:** 64
2. **Test Accuracy:** 63.94 %
3. **Learning rate:** 0.0001
4. **Epochs:** 5

#### **iii. Mini Batch with momentum SGD**

1. **Batch size:** 64
2. **Momentum:** 0.9
3. **Test Accuracy:** 75.79 %
4. **Learning rate:** 0.0001
5. **Epochs:** 5

#### **iv. Mini Batch Adam**

1. **Batch size:** 64
2. **Test Accuracy:** 82.33%
3. **Learning rate:** 0.0001
4. **Epochs:** 12

### **Conclusion:**

- *On Comparing Test Accuracies we can say that the model trained with Adam optimizer is more efficient than model trained with other optimizers.*



**(b)** As we get best Test Accuracy with above model trained with Adam, so tried for various mini batches for this model only.

Mini-Batch Size	Test Accuracy	Gradient update plot
16	82.85 %	
32	81.89 %	
64	82.33 %	

### Conclusion:

- As we observe from above table that there is good gradient flow even in the starting layer (conv layers) too in any of the batch size taken. But for comparison basis from the above table we can observe that we achieve comparatively good Test Accuracy with model having batch size 16 followed by 64 batch size and the 32 batch size.

- Also the reason for above point can be that in the model with batch size 16 has good gradient flow while backpropagating derivate towards initial layers also (conv layer).

### (c) Advantage of shuffling and partitioning the mini batches

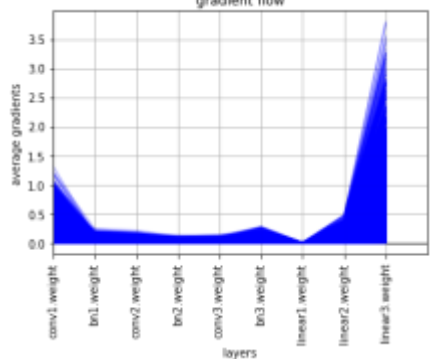
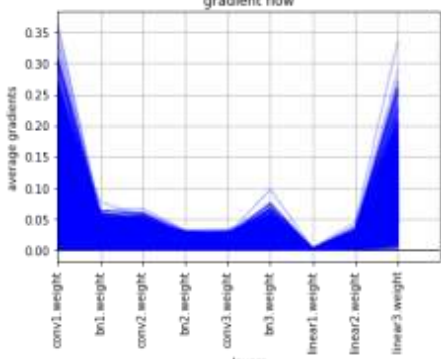
- Shuffling the dataloader has major advantage that shuffling leads to disrupt the model from learning the sequence of input coming. For example if we pass data in the form like first all cat images are given as input then all dog images as input. Then model may learn the sequence of coming of data and can perform poorly while testing. So model needs to see shuffled data of all classes during its learning/training phase.
- Partitioning the dataset into mini-batches also has a major advantage that during training time weight updation happens after every batch instead of the brute force training where weight updation happens after the whole dataset is passed once(i.e. after every epoch). Training model with mini-batches can improve the learning of the model by learning features by doing weight updation after every batch.

(d) Tries with various beta ( $\beta_1$ ,  $\beta_2$ ) values for Adam optimizer, As betas are used for computing running averages of gradient and its square.

$\beta_1$ : used to decay running average of gradient.

$\beta_2$ : used to decay running average of square of gradient.

Mini-Batch Size	Test Accuracy	Gradient update plot
$\beta_1 = 0.0$ , $\beta_2 = 0.0$	79.13 %	

$\beta_1 = 0.2, \beta_2 = 0.2$	81.41 %	
$\beta_1 = 0.9, \beta_2 = 0.999$	82.33 %	

## Conclusion:

- As we observe from above table that there is good gradient flow even in the starting layer (conv layers) too in any of the model with various beta values taken. But for comparison basis from the above table we can observe that we achieve comparatively good Test Accuracy with model having  $\beta_1 = 0.9, \beta_2 = 0.999$ .
- Also the reason for above point can be that in the model with  $\beta_1 = 0.9, \beta_2 = 0.999$  has good gradient flow while backpropagating derivate towards initial layers also (conv layer).

## (e) Advantages of using Adam over other optimization methods

- Yes there are some advantages of using Adam over other optimizers. But sometimes we can see in some situations Mini-batchSGD with momentum performs better than Adam optimizer.
- For this dataset we can observe on the basis of Test Accuracy reported in part(a) tells that Adam performs better than any other optimizer because Adam refers to Adaptive moment estimation as it takes the advantage of combination of best parts introduced in

*RMSprop and SGD with momentum for variable learning rate, which helps in speeding the convergence rate and getting out if stuck in local minima. But for general purpose we can't say Adam always outperforms other optimizers as SGD with momentum can do slower convergence speed but can also do better convergence than Adam in some situations.*