Image Classification

Perumadura De Silva

Content

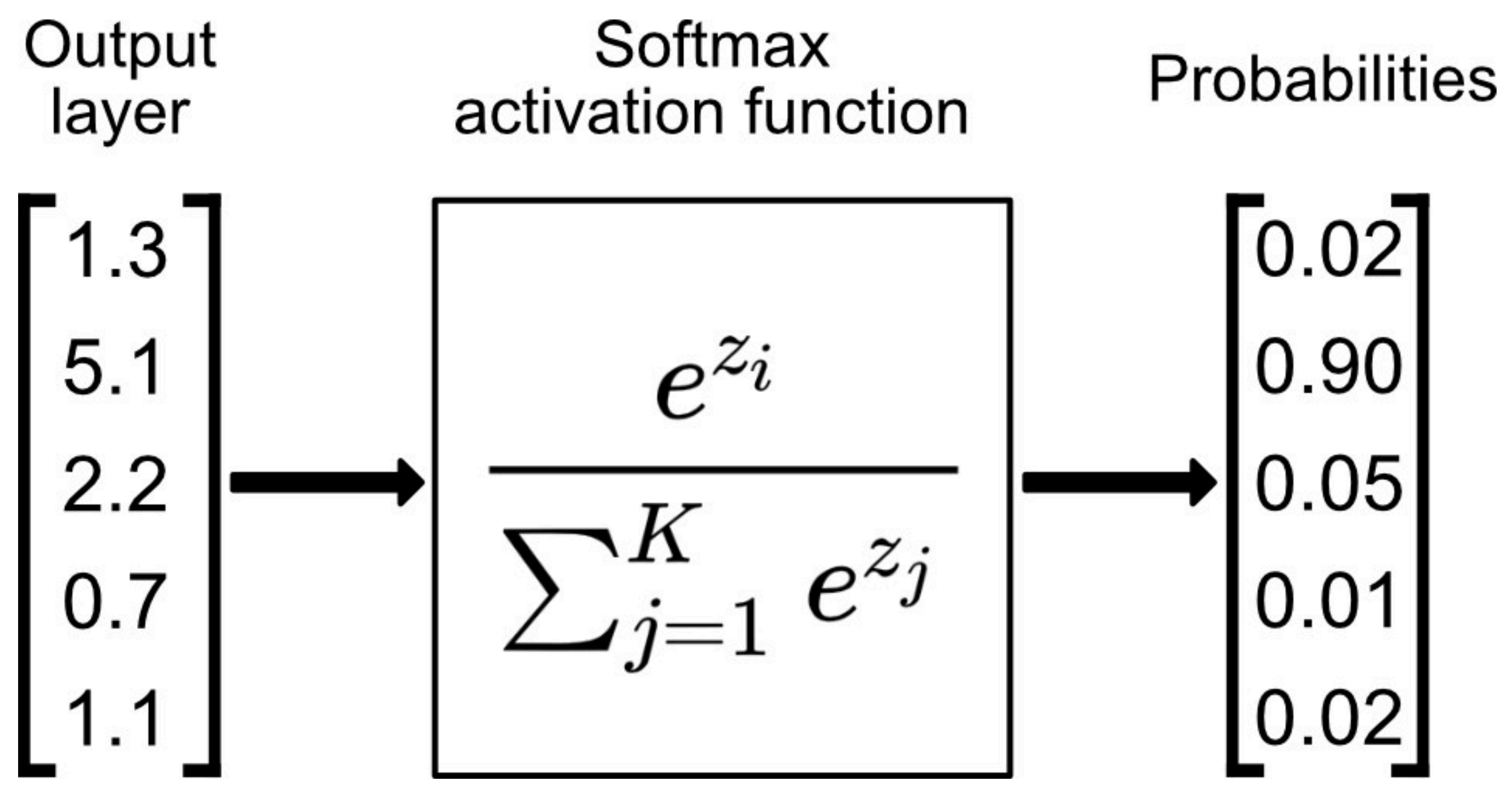
- Binary and multi-class classification
- Softmax function
- Cross entropy loss function
- Confusion matrix
- Accuracy, Precision and Recall
- Receiver Operating Characteristic (ROC) curve
- Transfer learning for classification
- Transfer learning experimental setup
- Experiment results and discussion

Binary and multi-class classification

- Binary:
  - Binary classification address the case where the label is either "True" or "False"
  - The most popular activation function used in binary classification is the sigmoid function:
    - $\sigma_{Sigmoid}(x) = \dfrac{1}{1 + e^{-x}}$
    - The sigmoid calculates the independent probabilities in multi class case
    - Sum of all sigmoid outputs does not necessarily adds up to 1.0
    - A solid decision on a prediction can be found by thresholding the outputs
      - $output = 1$ if $prob > threshold$ else 0

Binary and multi-class classification

- Multi-class
  - Multi-class addresses the case with 2 or more classes
  - The activation function used in this case is the Softmax function
  - A solid decision on the predicted class is made by taking the class that max the probability
    - $output = argmax_{i \in [i...N]}(\sigma_i^{Softmax}(z))$; z: inputs (logits) and N: Number of classes



Ref.0: Conversion of logits to Softmax probabilities

Softmax function

- A deep learning model at the output layer, outputs classification logits z (unnormalized)
- The Softmax function normalizes these logits to a discrete probability distribution
- The probabilities of the logits are not independent
- $\sigma_i^{Softmax}(z) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$
- where z is the output logits and i is the class we consider out of N classes
- All probabilities of the Softmax output sums to 1

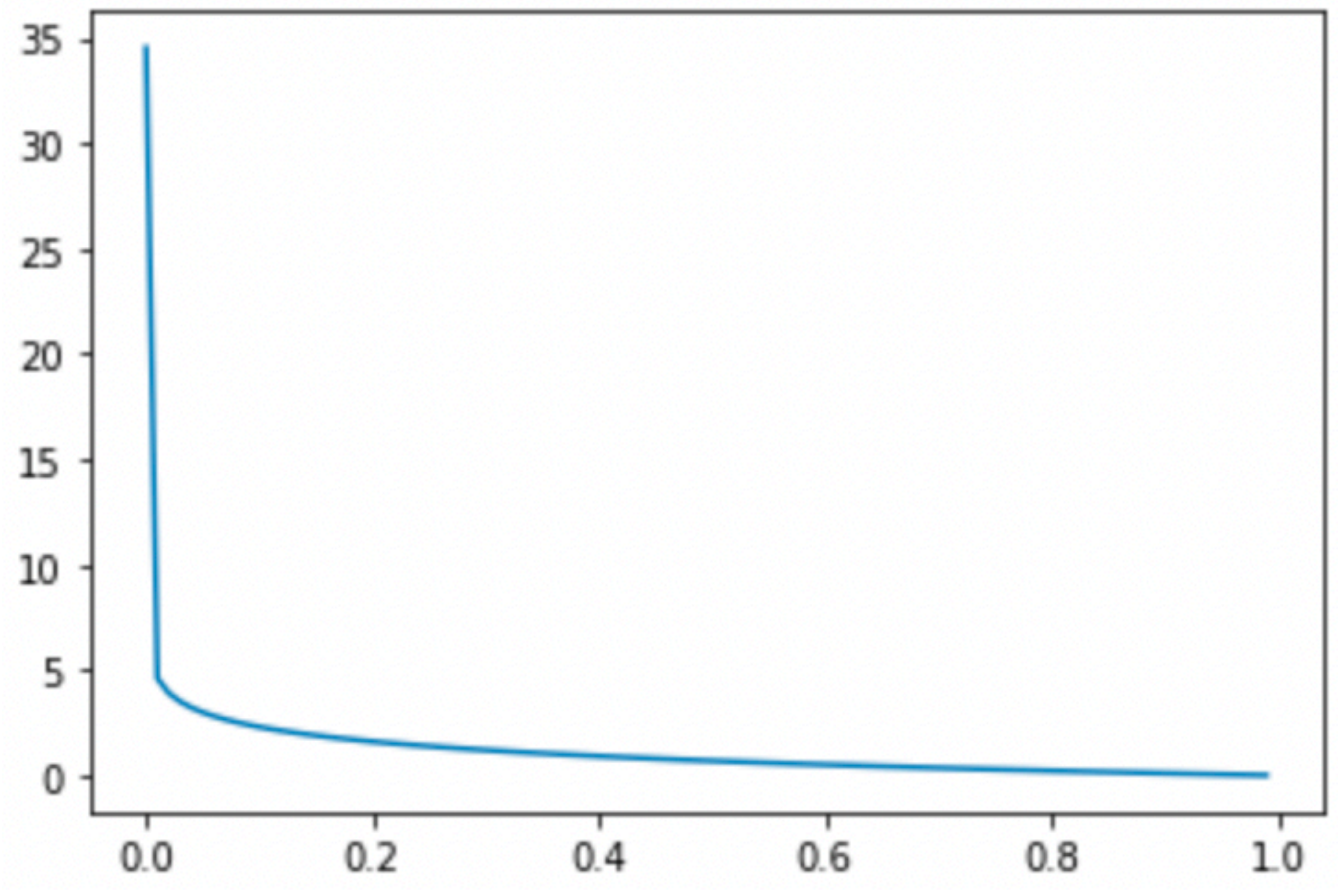
```
logits = [0.3, 1.6, 3.1, 5.6]
softmax = lambda z, zs: np.exp(z) / sum(np.exp(zs))
norm_logits = list(map(lambda z: softmax(z, logits), logits))
print(norm_logits)
print(sum(norm_logits))
```

```
[0.004515676666643495, 0.016569357344960955, 0.07425870771543724, 0.9046562582729584]
1.0
```

Cross entropy loss function

- In a discreet or continuous probability distribution:
  - The information quantifies the uncertainty of a given event
  - Deterministic events carry very less information (uninformative) (e.g. Sun rose today)
  - Less likely events carries more information (e.g. Today there was an eclipse)
  - Independent events should accumulate information
  - Information (self) of an event x: $I(x) = -log(P(x)) = log\left(\frac{1}{P(x)}\right)$; $P(.)$ probability mass function
  - When log is base e, the unit of information measurement is $nats$
  - $bits$ or $shannons$ for the base 2

Information



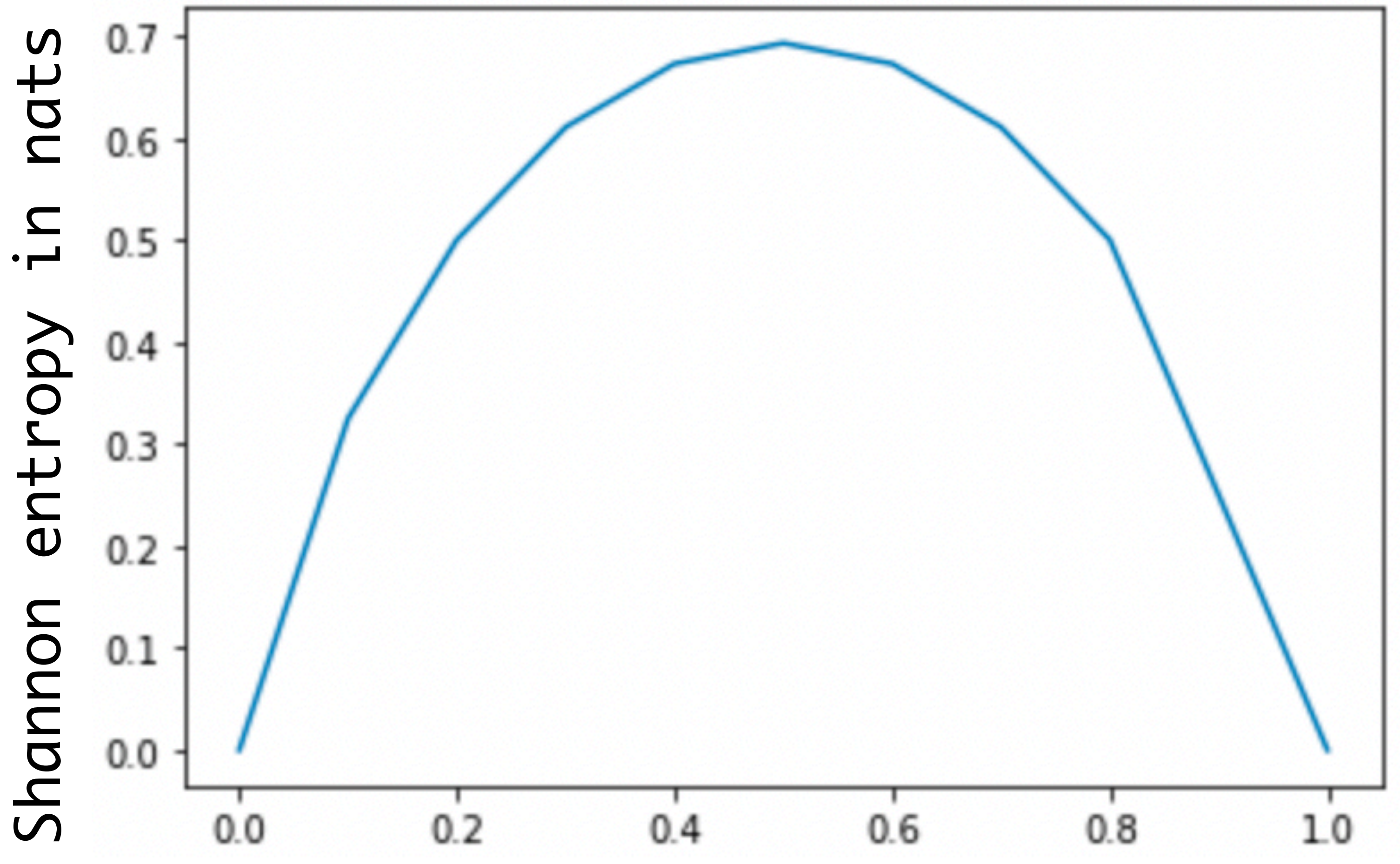
Probabilities $P(x)$

The plot shows that, an event that is guaranteed not to happen $P(x) \approx 0$ has the highest uncertainty and deterministic events $P(x) = 1.0$ has no information (uncertainty)

Check Ref-3 for more details

Cross entropy loss function

- The entropy quantifies the uncertainty of a given distribution (expected amount of information)
- Near deterministic distributions have zero entropy and uniform distributions have very high entropy
- Entropy (amount of information) of a discrete distribution:
  - $H(x) = \mathbb{E}_{x \sim P}[-log(p(x))] = -\sum_{i=1}^{K}[P(x_i)log(P(x_i))]$; K: Number of possible outcomes
- Taking the same idea, cross entropy measures the difference between a true distribution $P(x)$ and an estimated distribution $Q(x)$
  - Cross entropy loss: $H(P_{true}, Q) = -\sum_{i=1}^{K} P_{true}(x_i)log(Q(x_i))$
  - Cross entropy is always higher than the entropy except for the case where $Q(x) \approx P(x)$



$P(X=1)$: probability a binary random variable is equal to 1

When a binary random variable is sampled from a Bernoulli distribution it can take a value 1 with a probability p and 0 with a probability (1-p);
$P(X=1) = p = 1 - P(X=0) = 1 - q$
So the entropy here is calculated by:
$entropy = -(1-p)log(1-p) - plog(p)$
Highest entropy when p = 0.5
Check ref-2 and 3

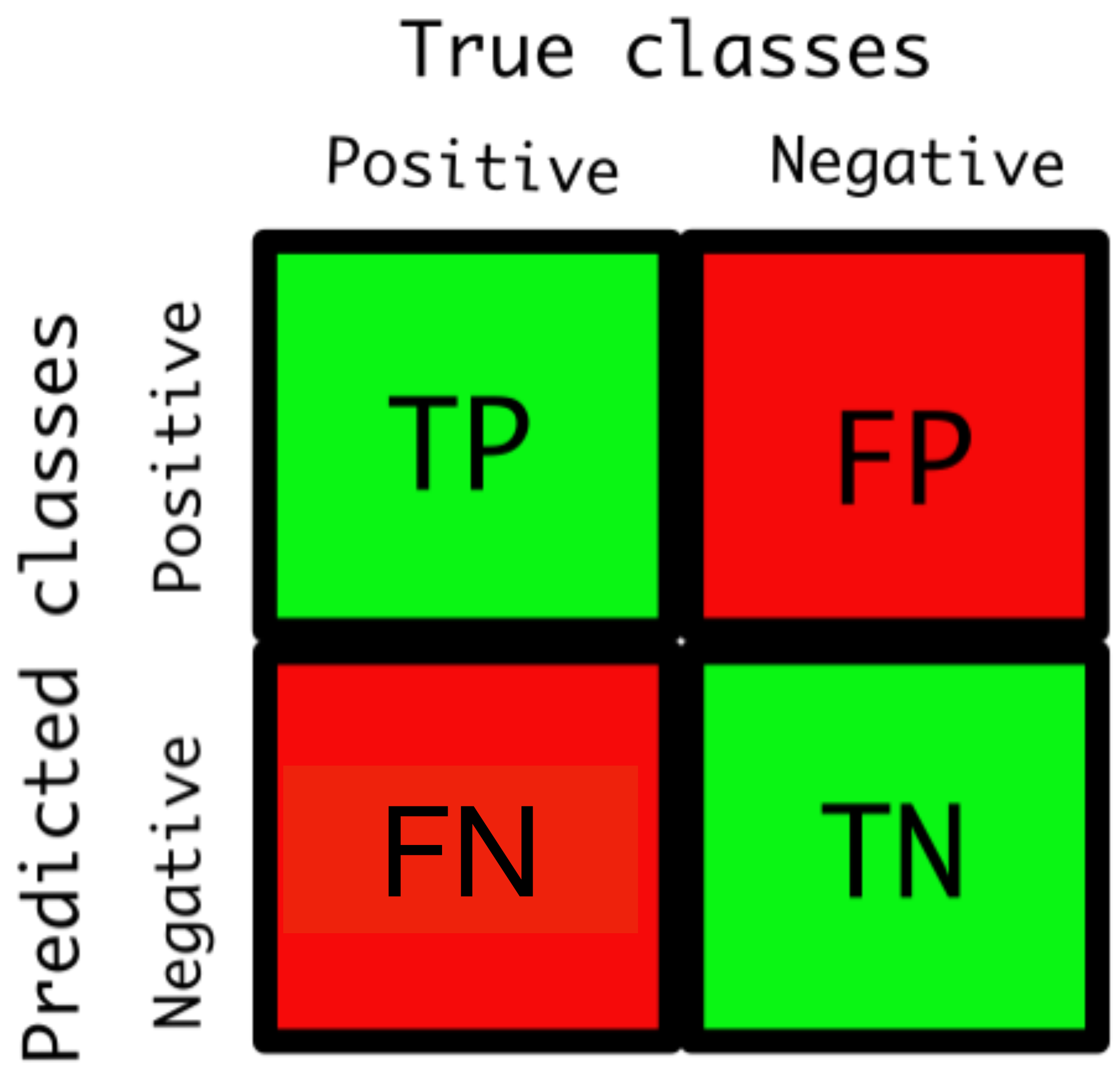
Check Ref-3 for more details

Cross entropy loss function

- Classification example:
  - There are 3 classes
  - Take: a model outputs given $class2$ image
  - $\sigma(z) = [class1 = 0.23 \quad class2 = 0.63 \quad class3 = 0.14]$ as normalised logits (Softmax probabilities)
  - The onehot encoded label is $[0,1,0]$
  - The cross entropy for this particular input:
    - $H = -0 * log(0.23) - 1 * log(0.63) - 0 * log(0.13) = 0.462$

Confusion Matrix (CM)

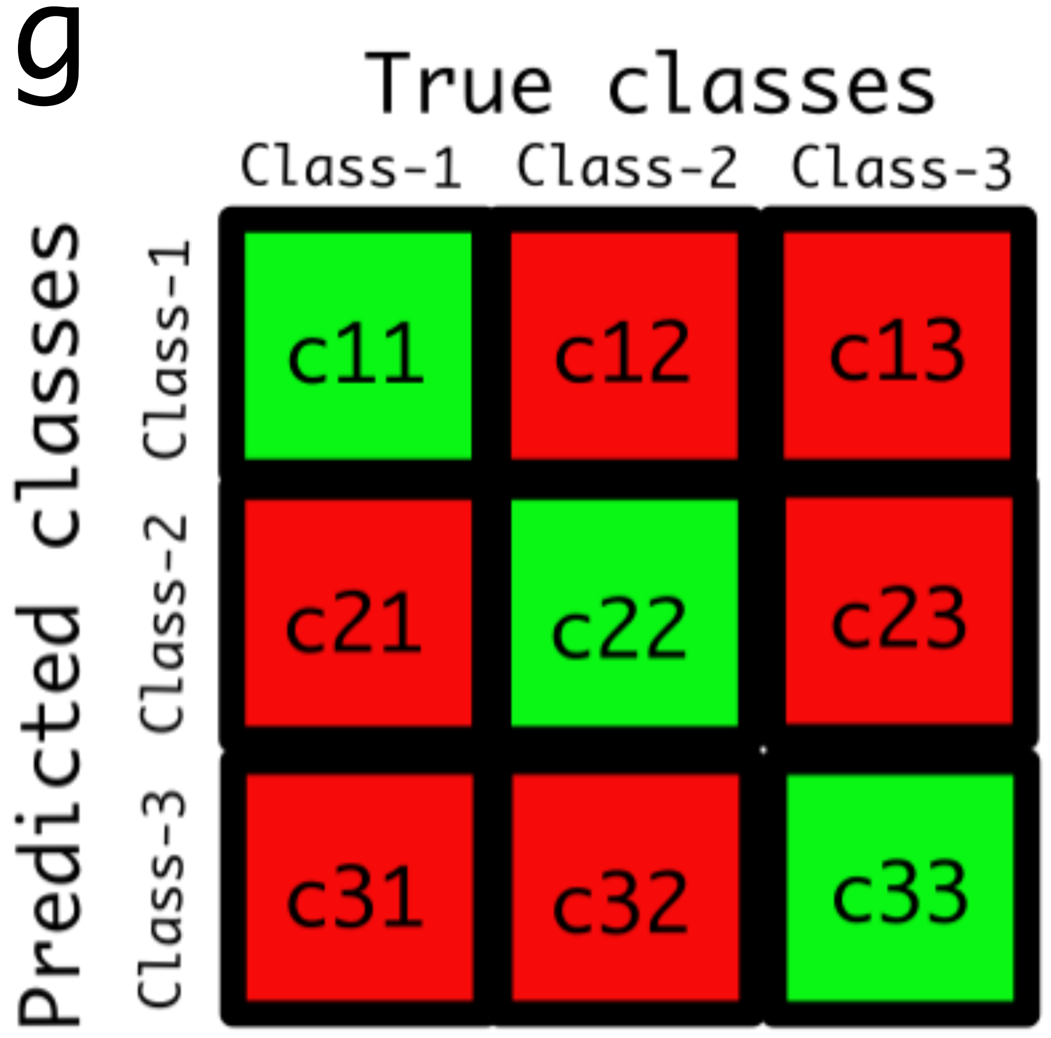
- The CM measures the performance of a classification model (binary or multi-class setup)
- It specifically shows the class confusion by means of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN)s
- Binary case:
  - TP: The prediction is positive and true (matches the label 1)
  - TN: The prediction is negative and true (matches the label 0)
  - FP: The prediction is positive but not true (not the label 1 but 0)
  - FN: The prediction is negative but not true (It is label 1 not 0)



Binary Confusion Matrix

Confusion Matrix (CM)

- Multi-class case:
  - Similar to the binary case, the Multi-class case considers all the classes when calculating TP, TN, FP and FNs
  - Eg

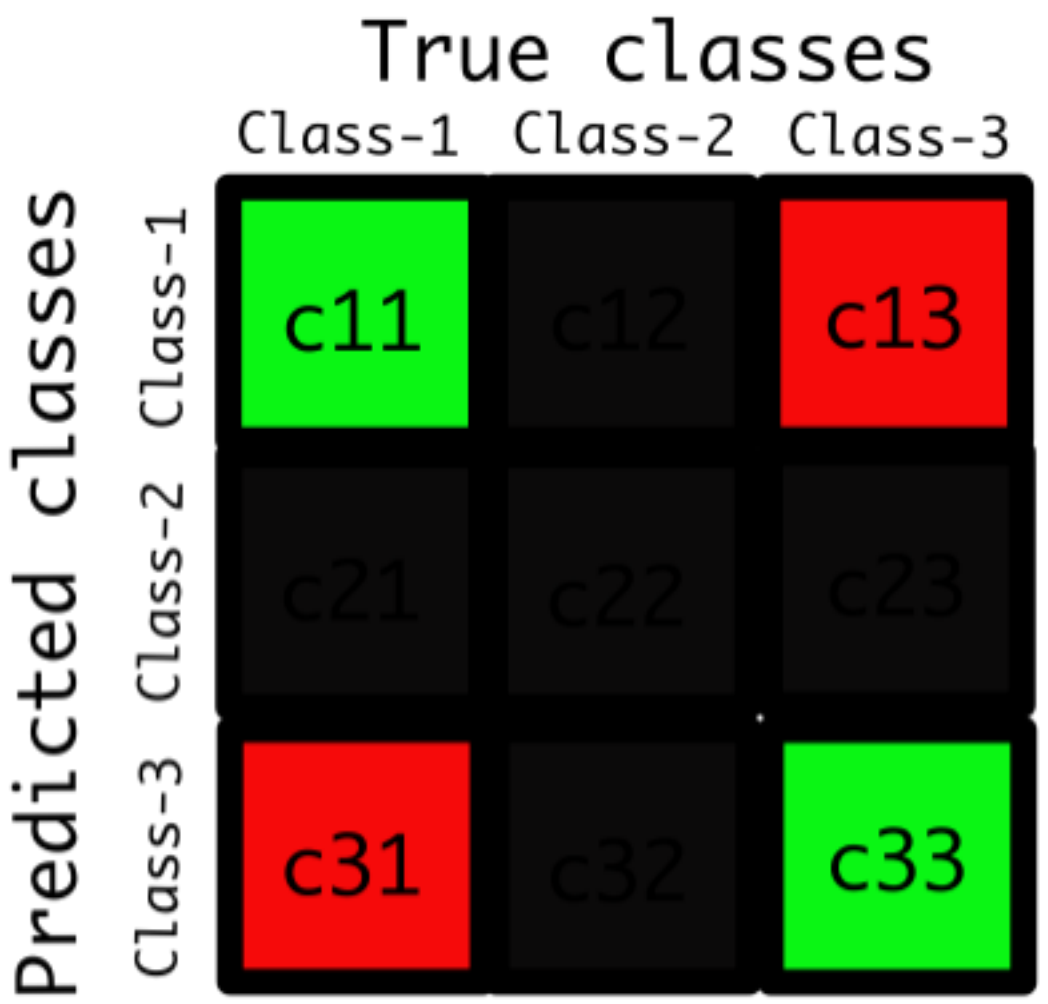


- True classes: What you have given to the NN to predict
- Predicted classes: The predictions of the corresponding True class inputs
- c11: number of class-1 predicted as class1
- c21: number of class-1 predicted as class2
- $TP_{total} = c11 + c22 + c33$

Confusion Matrix (CM)

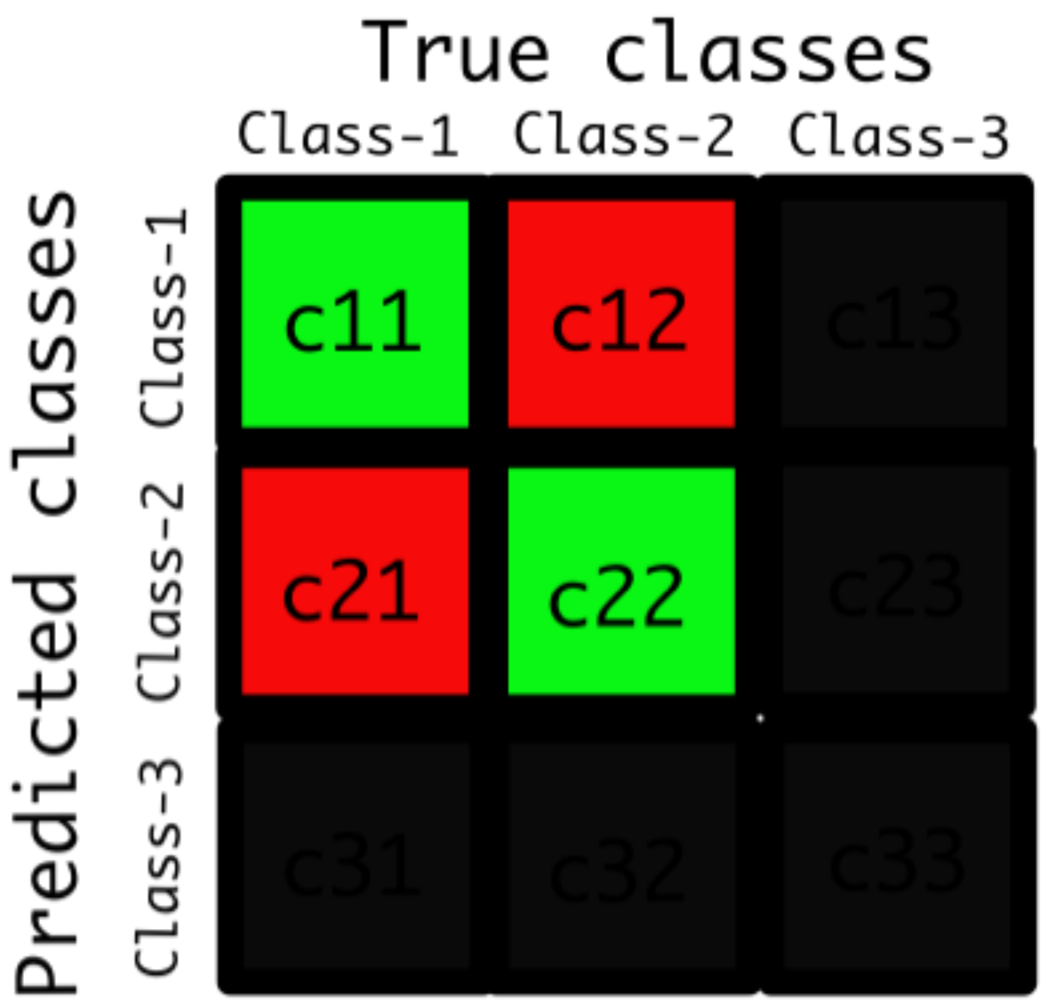
True Negative class-1 case:

$$TN_{class-1} = c22 + c23 + c32 + c33$$



True Negative class-1 case:

$$TN_{class-2} = c11 + c13 + c31 + c33$$



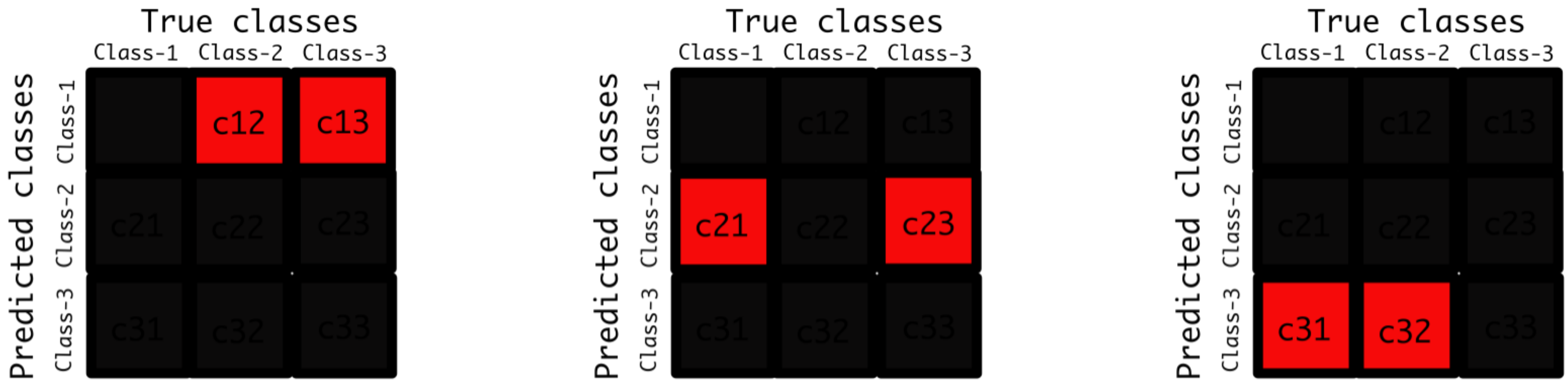
True Negative class-1 case:

$$TN_{class-2} = c11 + c12 + c21 + c22$$

True Negatives: The samples that does not truly belong to the subjected class

$$TN_{Total} = TN_{class-1} + TN_{class-2} + TN_{class-3}$$

Confusion Matrix (CM)



False positive class-1 case:
$FP_{class-1} = c12 + c13$

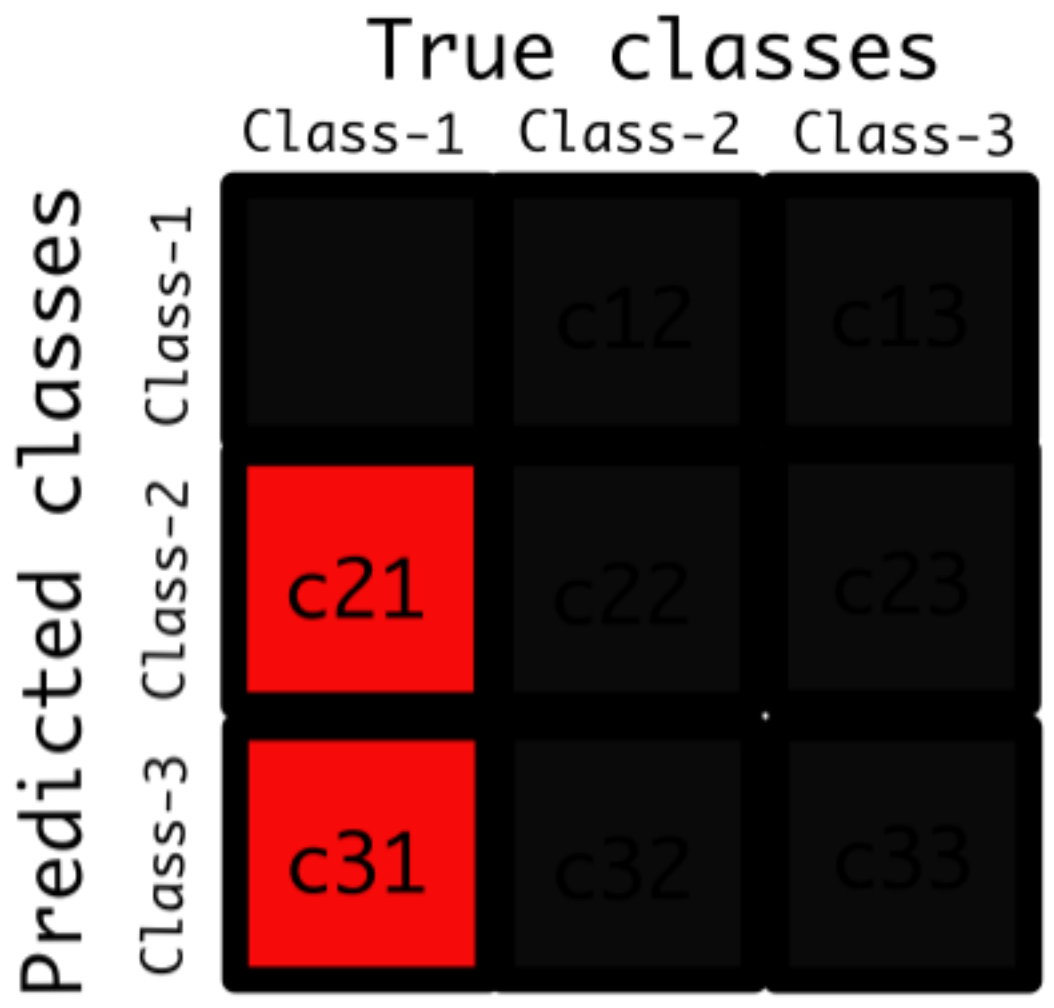
False positive class-2 case:
$FP_{class-2} = c21 + c23$

False positive class-2 case:
$FP_{class-3} = c31 + c32$

False positive: The samples that are predicted wrong for a given class

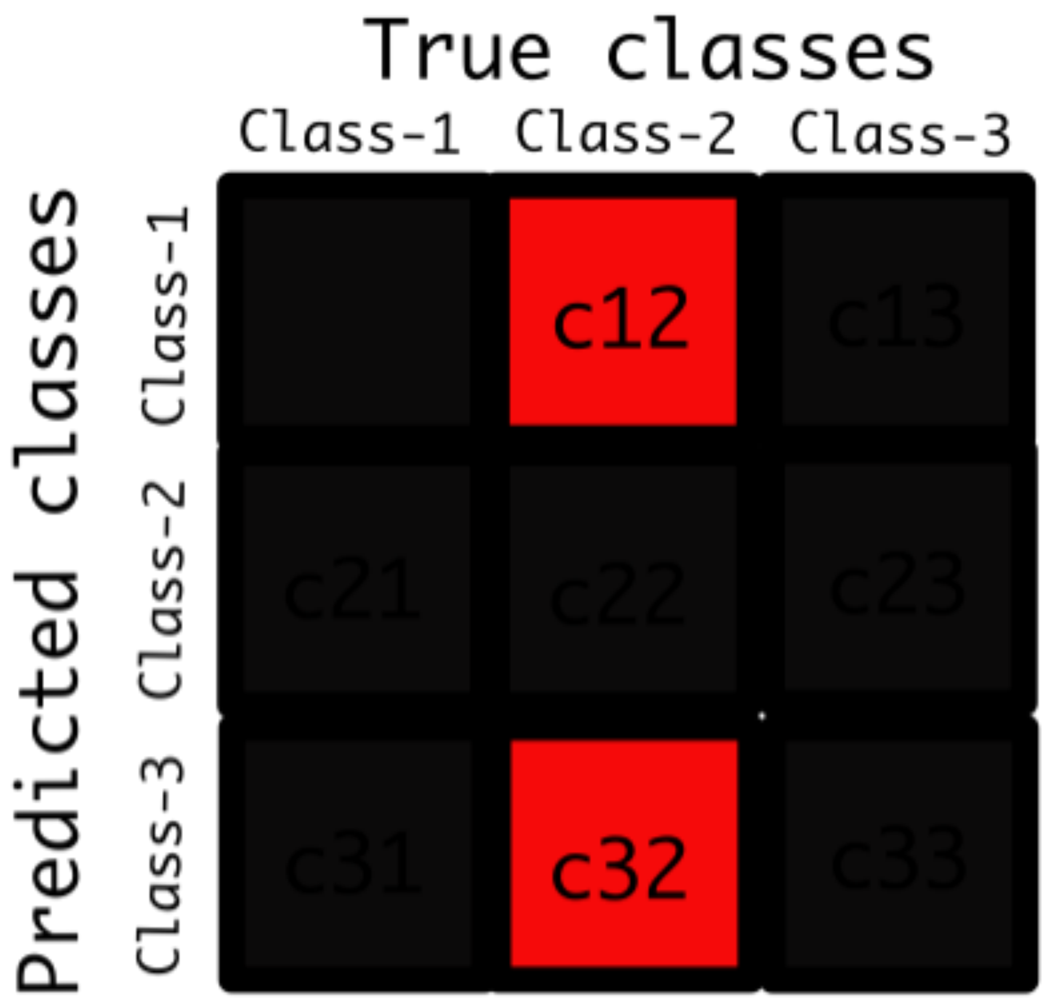
$$FP_{Total} = FP_{class-1} + FP_{class-2} + FP_{class-3}$$

Confusion Matrix (CM)



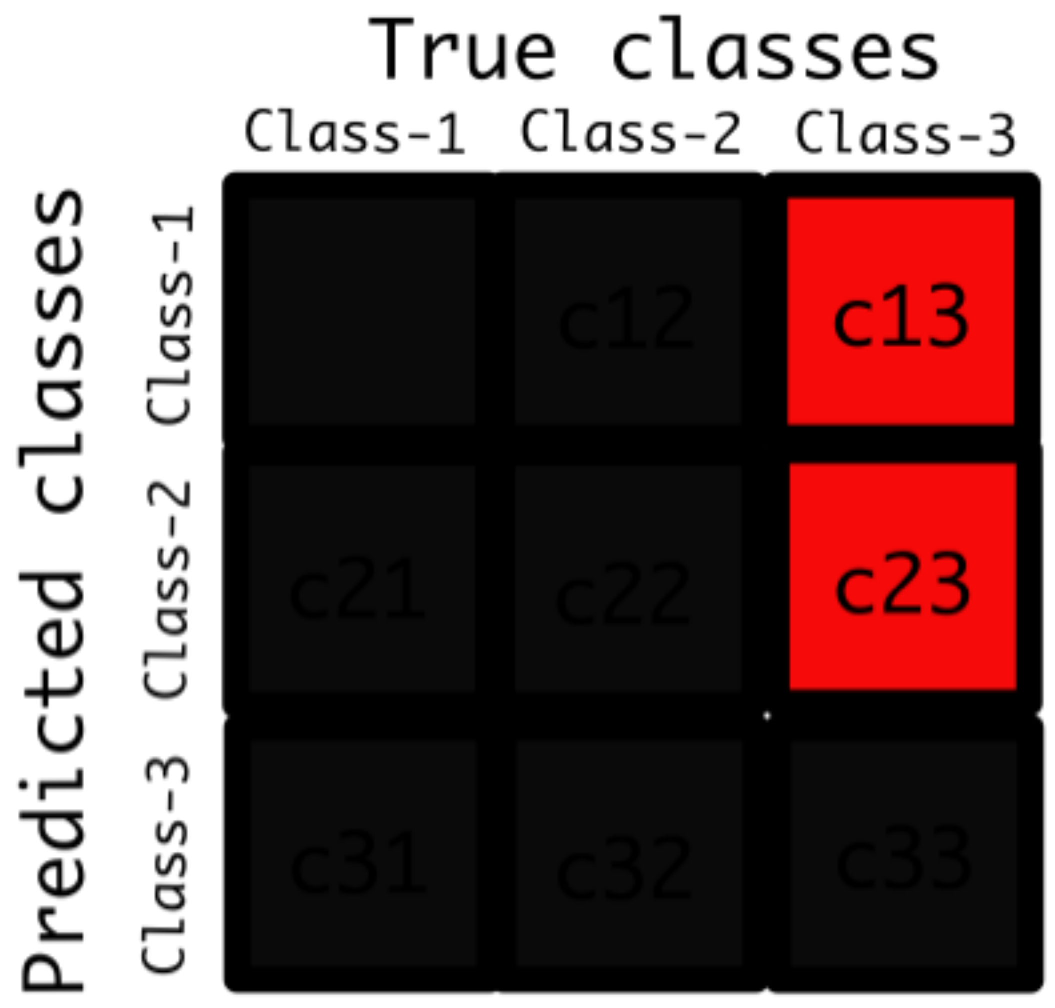
False Negatives class-1 case:

$FN_{class-1} = c21 + c31$



False Negative class-2 case:

$FN_{class-2} = c12 + c32$



False Negative class-3 case:

$FP_{class-3} = c13 + c23$

False Negatives: The samples that are predicted as wrong class

$$FN_{Total} = FN_{class-1} + FN_{class-2} + FN_{class-3}$$

Accuracy, Precision and Recall

- Accuracy
  - The model accuracy simply is the ratio between the total number of correct classifications and total number of classification
  - $Accuracy = \frac{TP_{total}}{TP_{Total} + TN_{Total} + FP_{Total} + FN_{Total}} = \frac{TP_{total}}{TOTAL}$ where $TOTAL$ is the total number of classifications
    - In above example
      - $TOTAL = c11 + c12 + c13 + c21 + c22 + c23 + c31 + c32 + c33$
  - Accuracy shows how often the classifier is correct in prediction
  - However accuracy does not reflect the true capability of a classifier (very sensitive to class imbalances in test set)

Accuracy, Precision and Recall

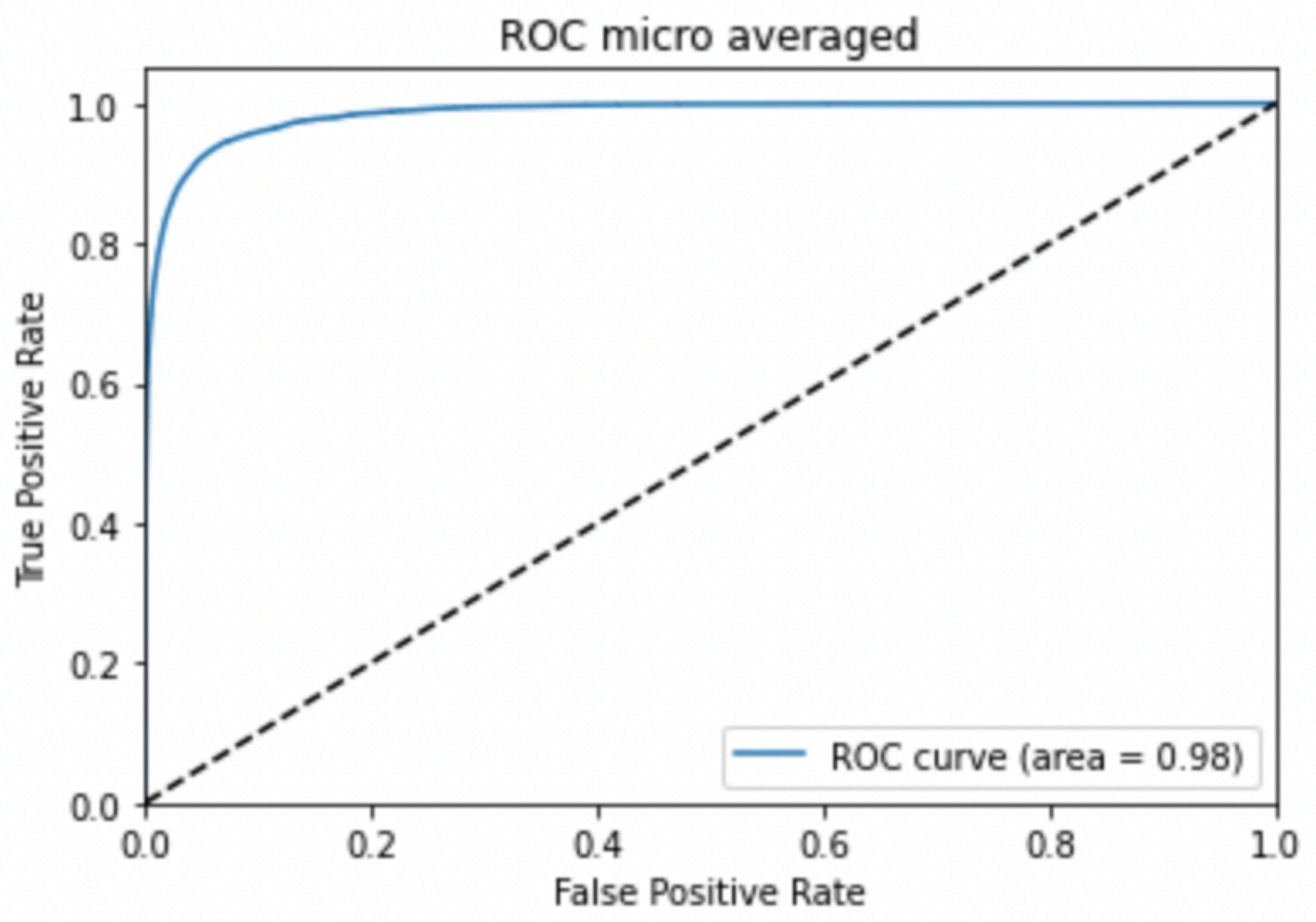
- Precision
  - Precision of a given class measures how well the predictions are, given the total number of predictions in that class
  - $Precision = \frac{TP}{TP + FP}$ Where $TP + FP$ are the total number of predictions in that class
- Recall
  - Recall quantifies the ability of a model to classify the relevant samples in a dataset. This gives an indication on the coverage of a given class
  - $Recall = \frac{TP}{TP + FN}$; $TP + FN$, relevant samples as in all the samples for a given class
  - Recall is also known as the sensitivity of the model
- Precision and recall can be related to each other inversely
- Specificity of a model: How many TN samples got correctly classified out of all negatives
  - $Specificity = \frac{TN}{TN + FP}$

Accuracy, Precision and Recall

- Average of Precision and Recall in multi-class classification
  - There are two types of averaging methods:
  - Macro average: $Precision_{macro} = \sum_{i=0}^{N} \frac{Precision_i}{N}$; N: number of classes
    - Gives equal importance to all the classes
    - Insensitive to the class imbalance
  - Micro average: $Precision_{micro} = \frac{TP_{total}}{TP_{total} + FP_{total}}$
    - Prefers the class with largest number of samples

Receiver Characteristic Operator (ROC) curve

- The ROC is originally an evaluation metric for binary classification
- ROC is plot of recall against the false positive rate (FPR) at different thresholds
- False Positive Rate (FPR): rate of negative samples classified as positive
  - $FPR = \dfrac{FP}{FP + TN} = 1 - specificity$
- It is used to find an optimal threshold that balances the recall and false positive rate
- The area under the ROC curve (AUC) measures the ability of a classifier to distinguish between the classes
  - Higher the better
  - Compare AUC of different classifiers to determine the best classifier for a given task



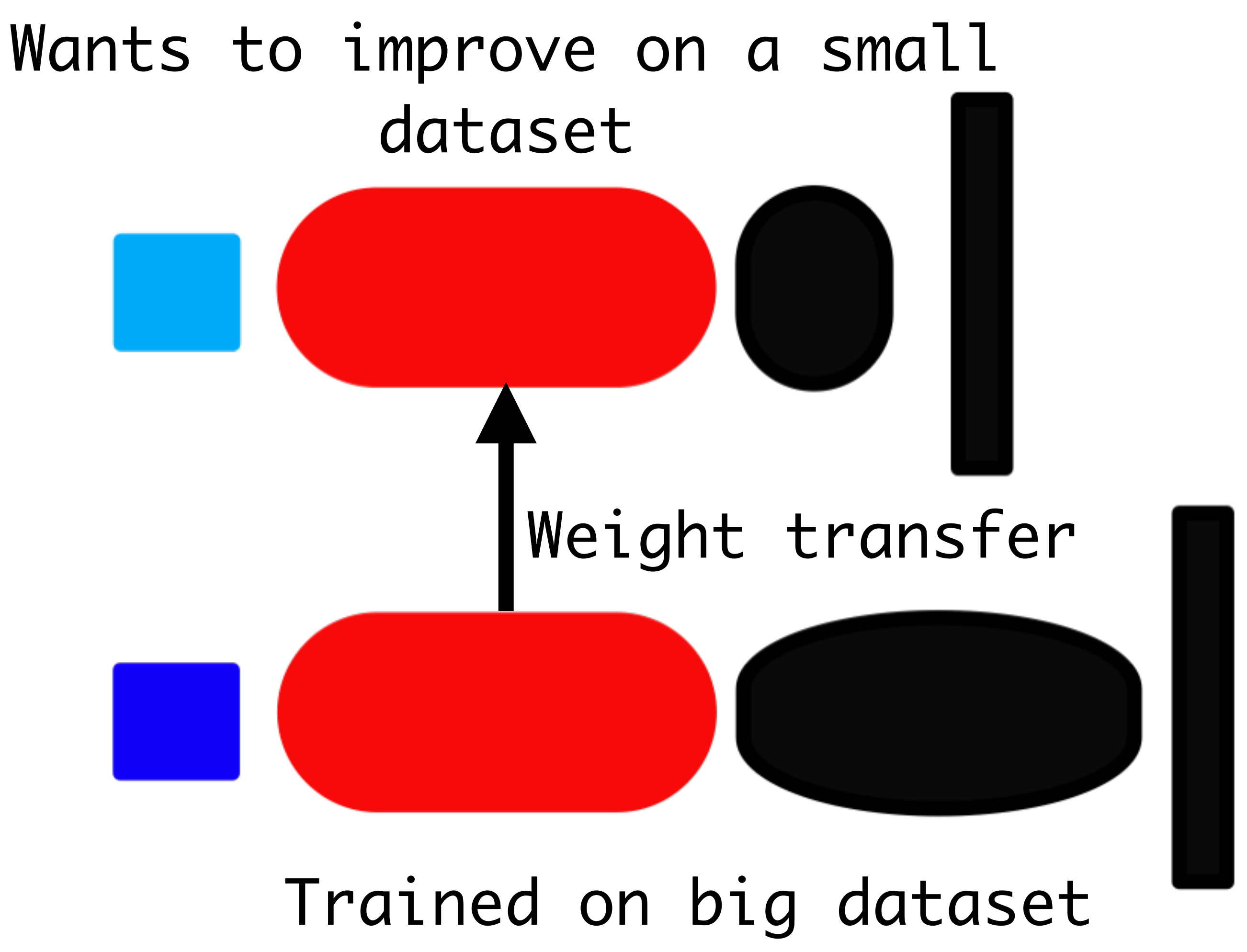
ROC curve example

Transfer learning in image classification

From the paper "How transferable are features in deep neural networks?"

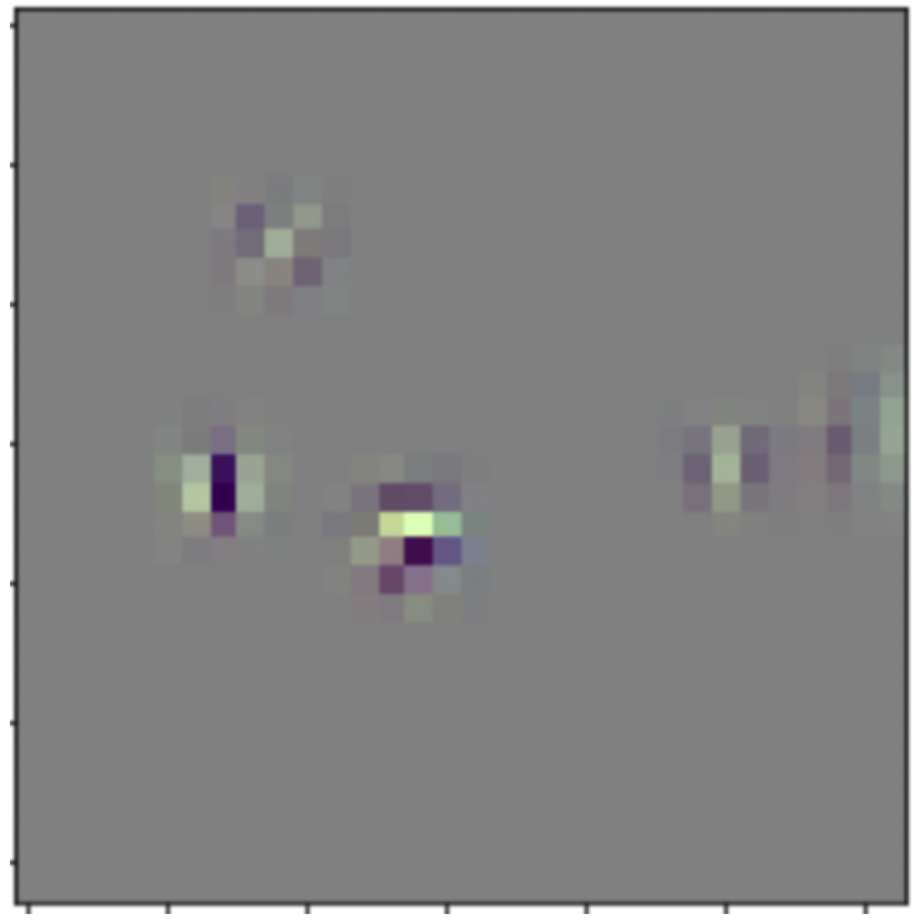
Transfer learning in image classification

- In the domain of deep learning, the transfer learning involves a source task (A) and an target task (B)
- The idea is to learn the features using a source task with large enough dataset and transfer the knowledge to the target task by means of learnt parameters
- The transferability between two tasks could depend on:
  - The distance between tasks (how similar they are)
  - The point of transfer (till which layer the knowledge is transferable)
  - Target dataset size (small dataset could lead to overfitting even in the transfer learning case)
- Once the point of transfer is determined (N^{th} layer), either the transferred features are frozen or fine tuned for the target task
- If the target dataset is very small and N layers have large number of parameter:
  - Fine tuning could lead to overfitting so leave it frozen
- Otherwise, fine tune to the new data

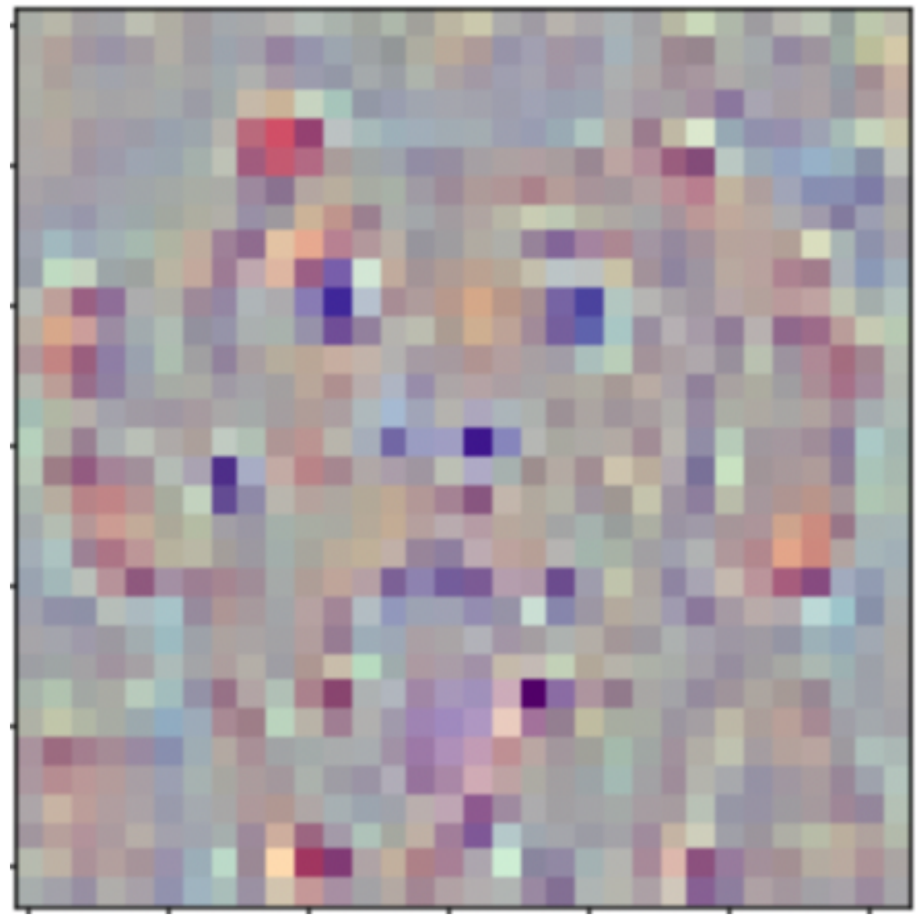


Transfer learning in image classification

- Generality and Specificity of a layer
  - Specific features: Features that are task specific
  - General features: Features that are common between tasks
  - The early layers (near input) of a deep convolutional neural network learns generalisable features such as color blobs, edges and corners
  - The later the layer is the generalisability drops and specificity increases (learns high level concepts such as full shapes)



General features



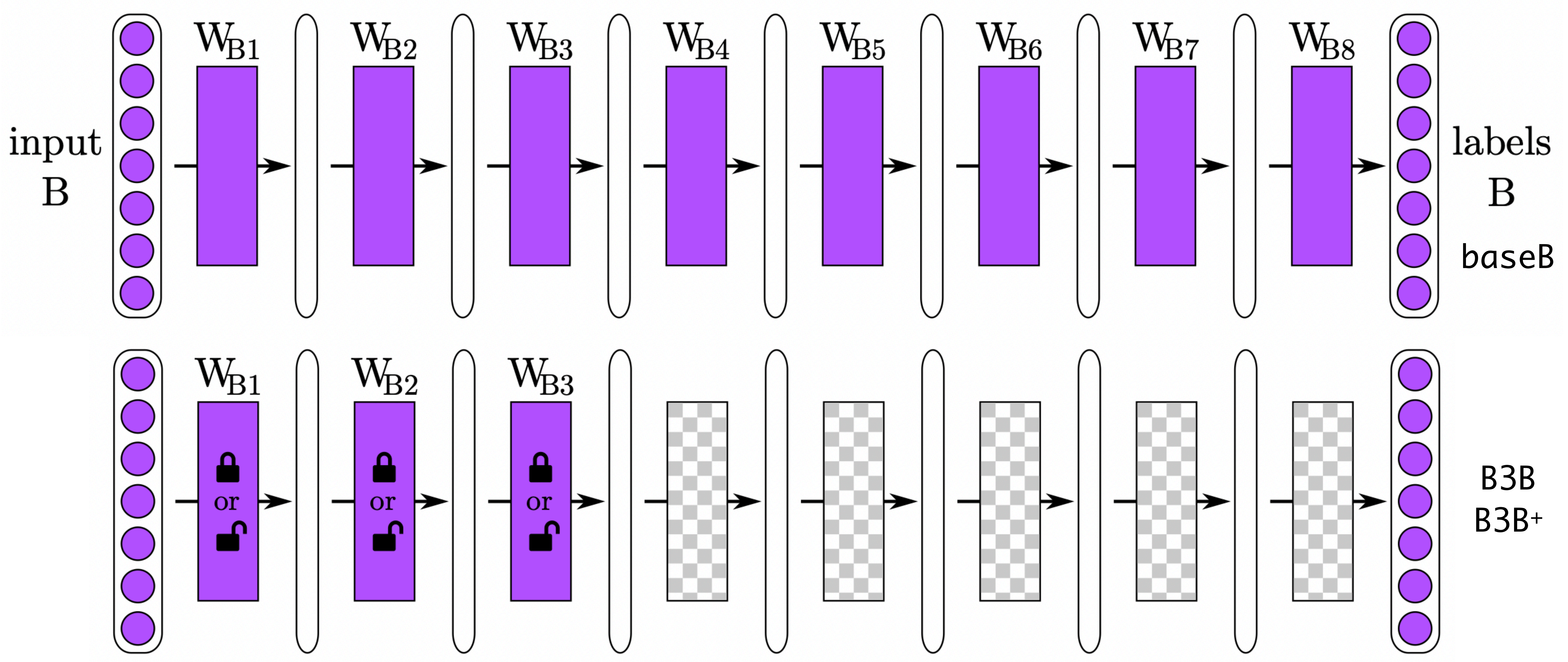
Task specific features

Transfer learning experimental setup

- Transfer learning experiment setup
  - Two tasks are considered:
    - source task A and target task B
  - Task A has large number of data and Task B has small number of data
  - Considered scenarios:
    - Scenario-1: Task A and B are very similar
      - Imagenet dataset split according to types of cats and dogs (similar domain)
    - Scenario-2: Task A and B are very different
      - Imagenet dataset split between man-made and natural images (different domains)
  - A Neural Network with 8 layers is used to train on both the tasks

Transfer learning experimental setup

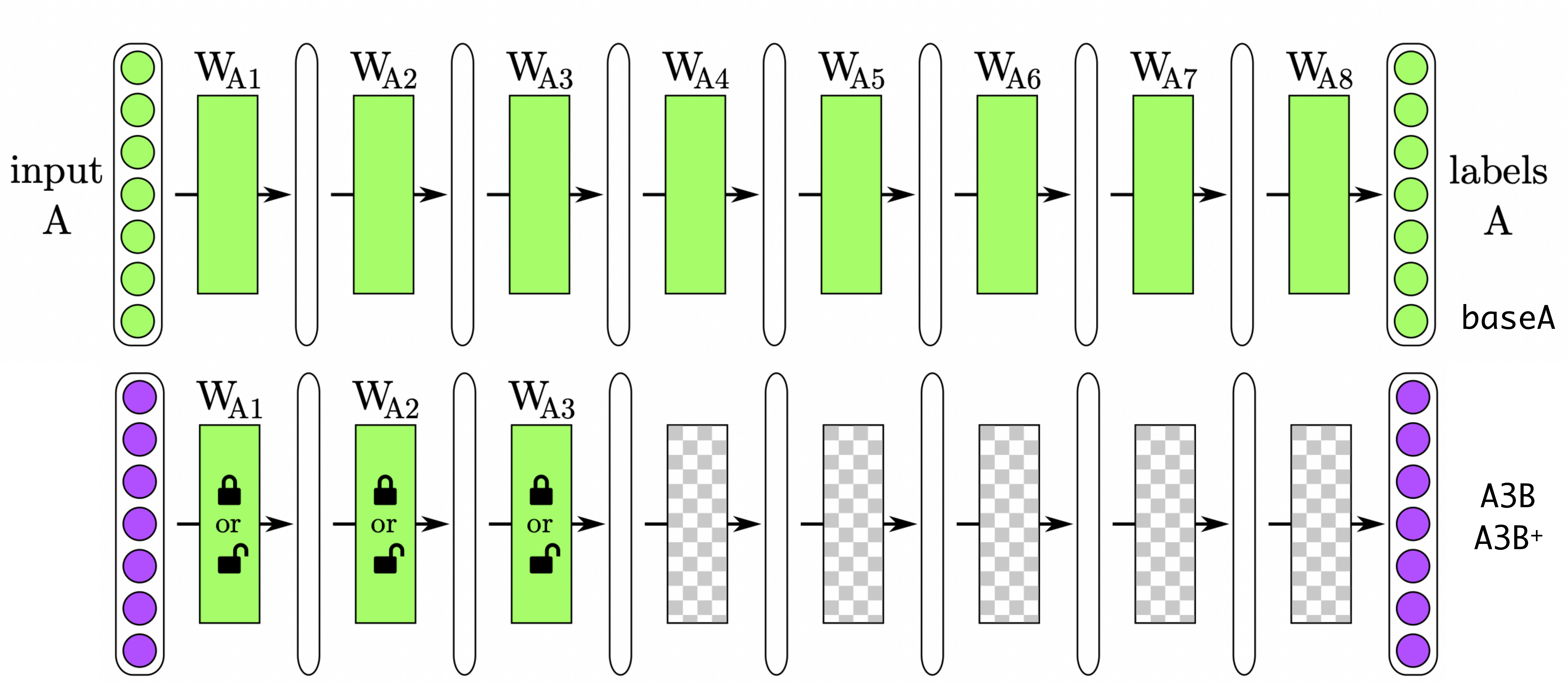
- Train setup-1:



Ref-1: Transferring within the same domain
baseB: Trained on the small dataset B (target)
B3B: Freeze 3 layers and randomly initialise the rest
B3B+: Fine tune the 3 layers together with the randomly initialised ones

Transfer learning experimental setup

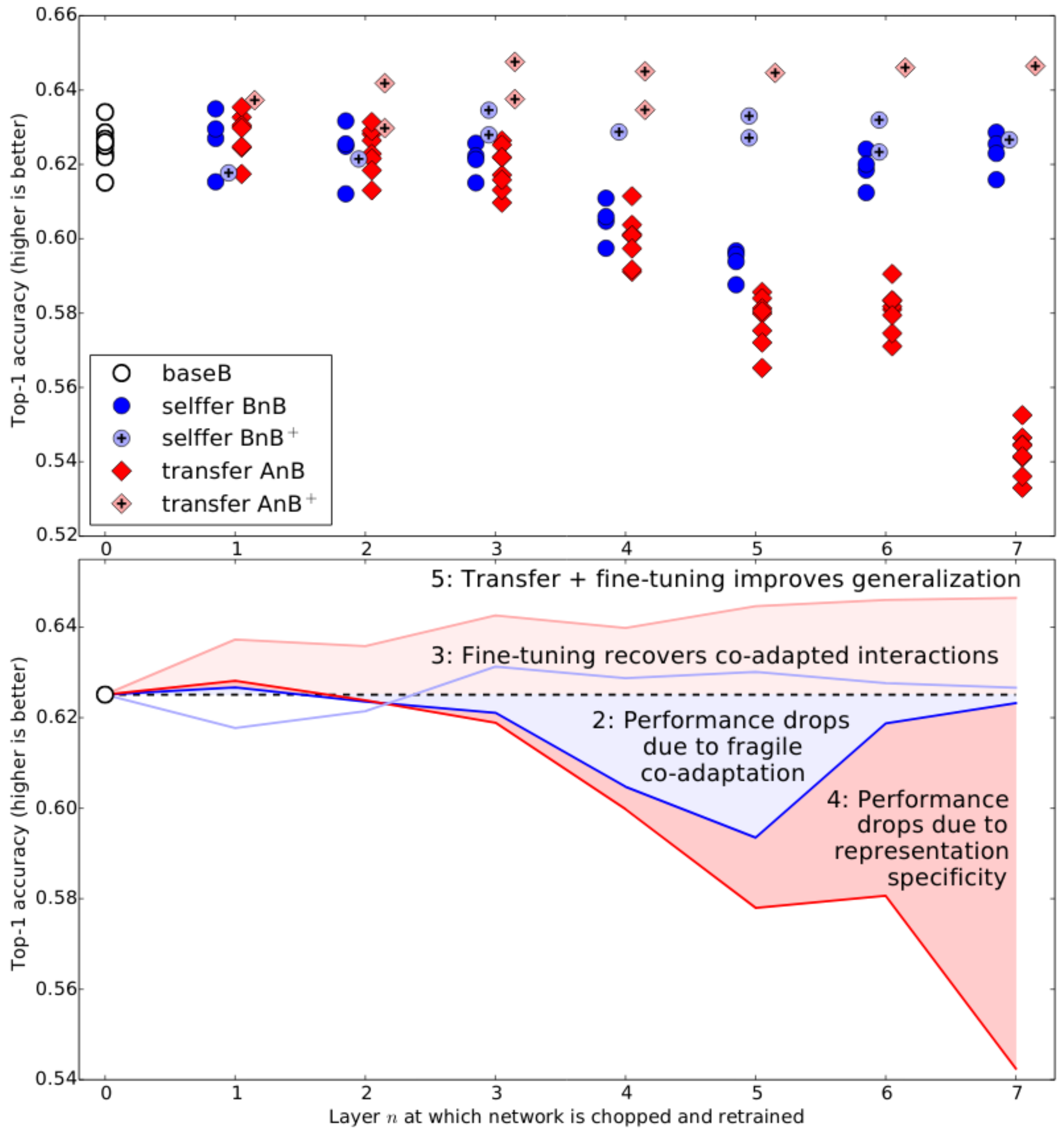
- Train setup-2:



Ref-1: Transferring from a domain with large number of data to small number of data
baseA: Trained on the large dataset A (source)
A3B: Freeze 3 layers and randomly initialise the rest
A3B+: Fine tune the 3 layers together with the randomly initialised ones

Experiment results and discussion

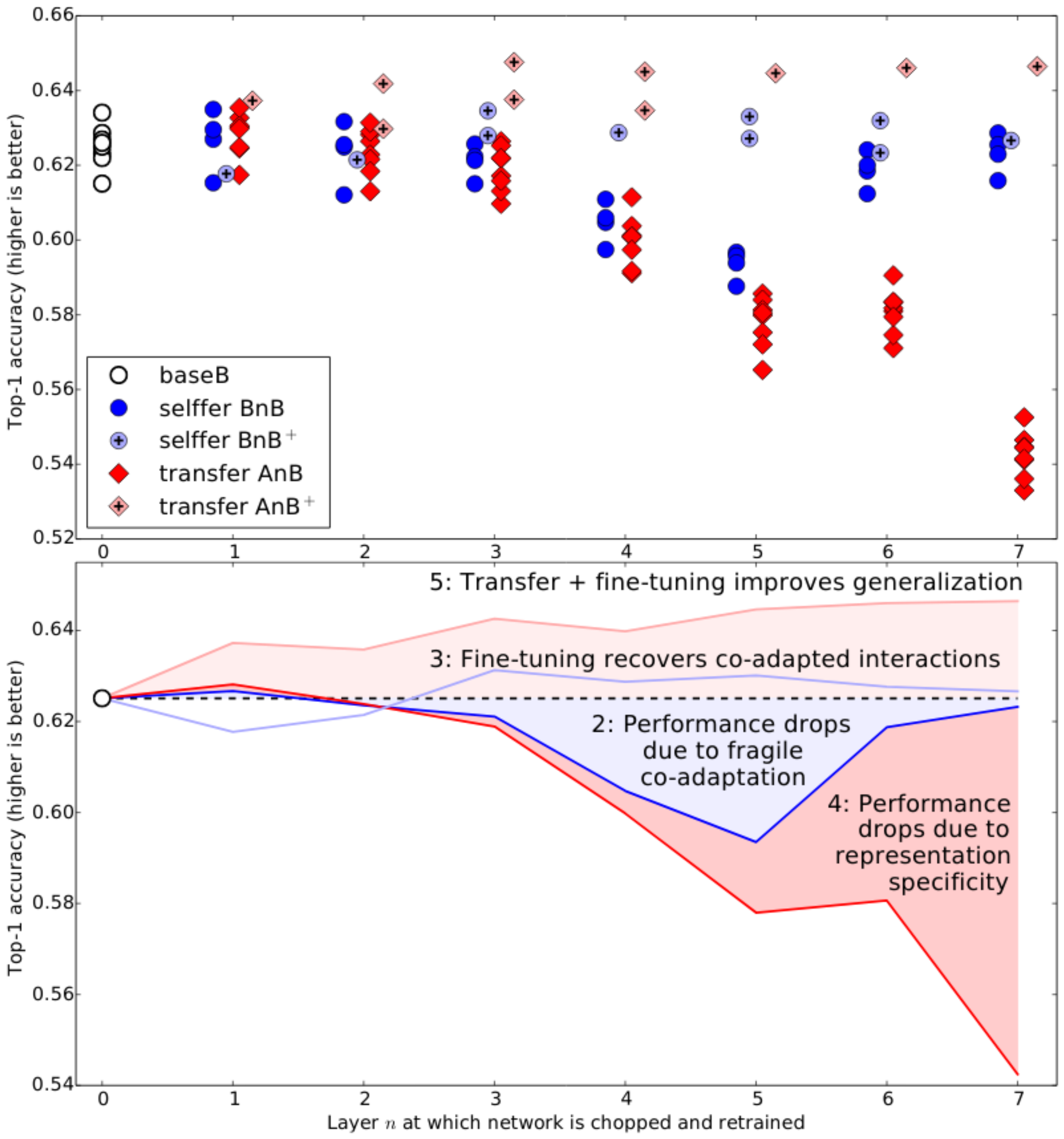
- Similar Domain case
- The following plots show the top-1 accuracies for the task B validation set (8 points: 8 A/B splits)
- White dots: The base accuracy for task B validation is 62.5% when only trained on task B
- Case: Adaptation within the same domain
  - Blue solid circle: Layers are kept frozen 🥶
    - The results here shows, that for the first 3 layers we have similar accuracies as expected (Because of generalisability)
    - But the drop after layer 4 indicates the co-adaptive features. The co-adaptive features are the features that depends/has a relationship to the proceeding layers
    - However, from layer 6-8 the accuracy again recovers indicating these layers are free of co-adaptive features



Ref-1: Performance plot, when transferring different trained layers

Experiment results and discussion

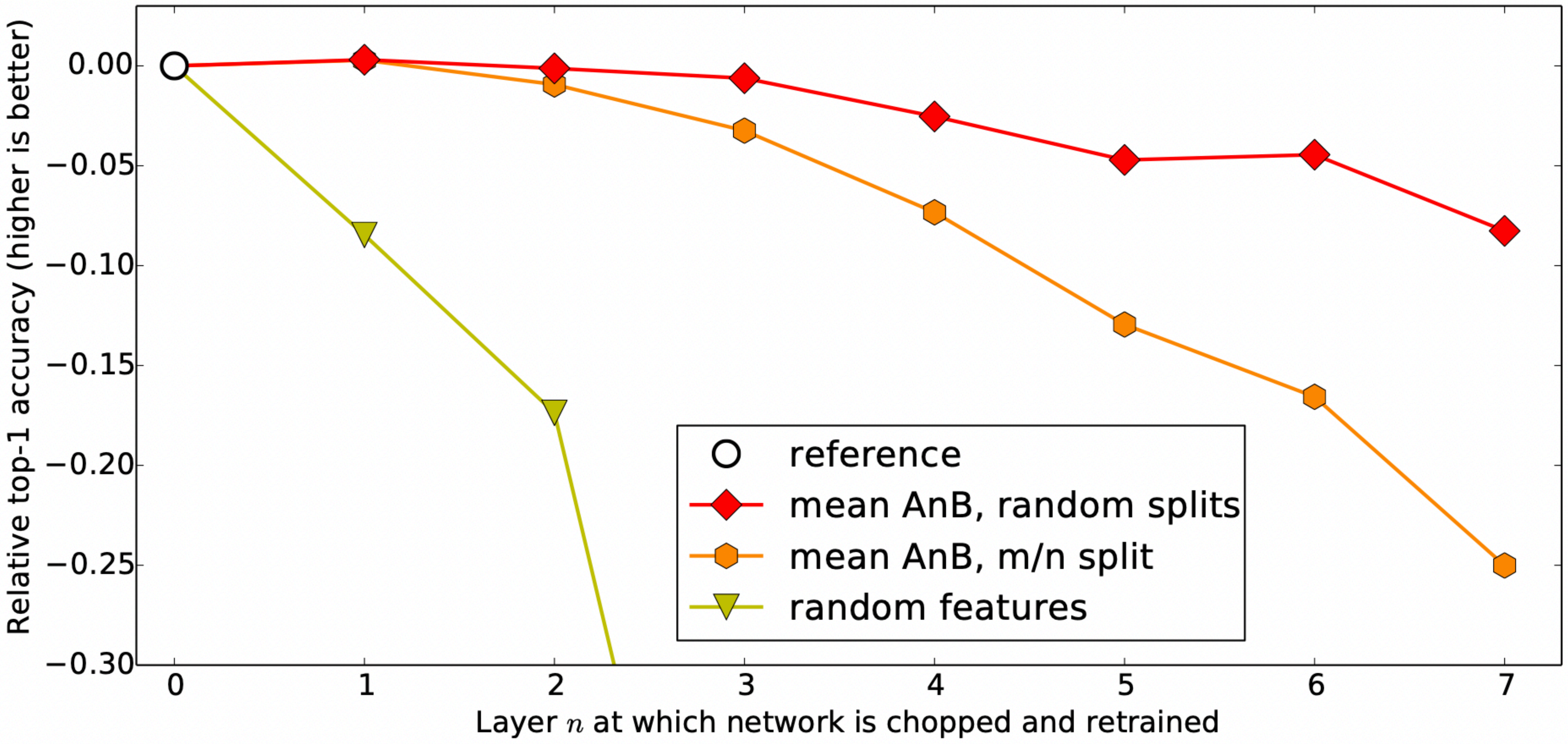
- Blue dot with a +: The frozen layers are also further trained along with the randomly initialised layers
  - The performance here is similar to the base accuracy since the scenario itself is similar to the task B only training (the base case)
- Red diamonds: Layers are transferred from task A to B and frozen 🥶
  - Here shows clearly the generalisability between tasks
  - The performance when only the first 3 layers are frozen gives similar accuracies as trained on task B itself (generalisability)
  - But after the layer 3 the performance drops significantly as the number of co-adaptive and specific features dominates the generalisable features
- Red diamonds with a +: Similar to the previous case, but the transferred features of task A further fine tuned
  - This results suggests that, when fine tuned after transferring from a different domain could lead to better generalisation compared to all the method discussed above
  - Be aware that depending on the size of the target dataset, there is a potential that this could lead to an overfitting



Ref-1: Performance plot, when transferring different trained layers

Experiment results and discussion

- Different domain case:



Ref-1: Transferring between dissimilar domains, The performance drops significantly.
A: man-made class data (source), B: natural class data (target)
The results are only for the AnB frozen weight case.

References

- Reference on Cross Entropy Loss:
  - Ref-0: https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60
  - Softmax derivative: https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/
  - http://machinelearningmechanic.com/deep_learning/2019/09/04/cross-entropy-loss-derivative.html
  - https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/
  - Ref-2: https://en.wikipedia.org/wiki/Bernoulli_distribution
  - Ref-3: CHAPTER 3. PROBABILITY AND INFORMATION THEORY; Deep Learning, An MIT Press book, Ian Goodfellow and Yoshua Bengio and Aaron Courville; https://www.deeplearningbook.org
- Transfer learning:
  - Ref-1:How transferable are features in deep neural networks?