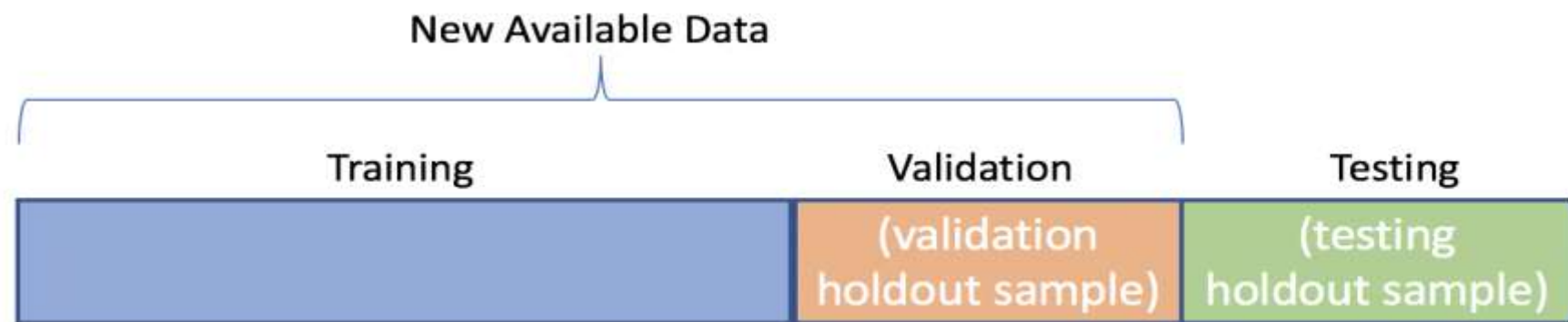
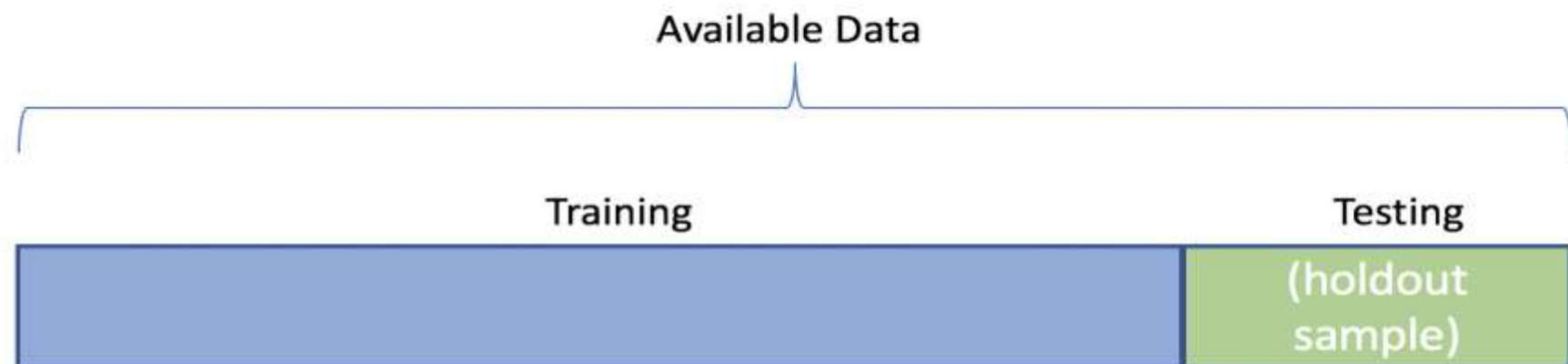


CNN

- **Training Dataset:** *The sample of data used to fit the model. The model sees and learns from this data.*
- **Validation Dataset:** *The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model **hyperparameters**. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.*
- **Test Dataset:** *The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.*

Dataset



To train the models

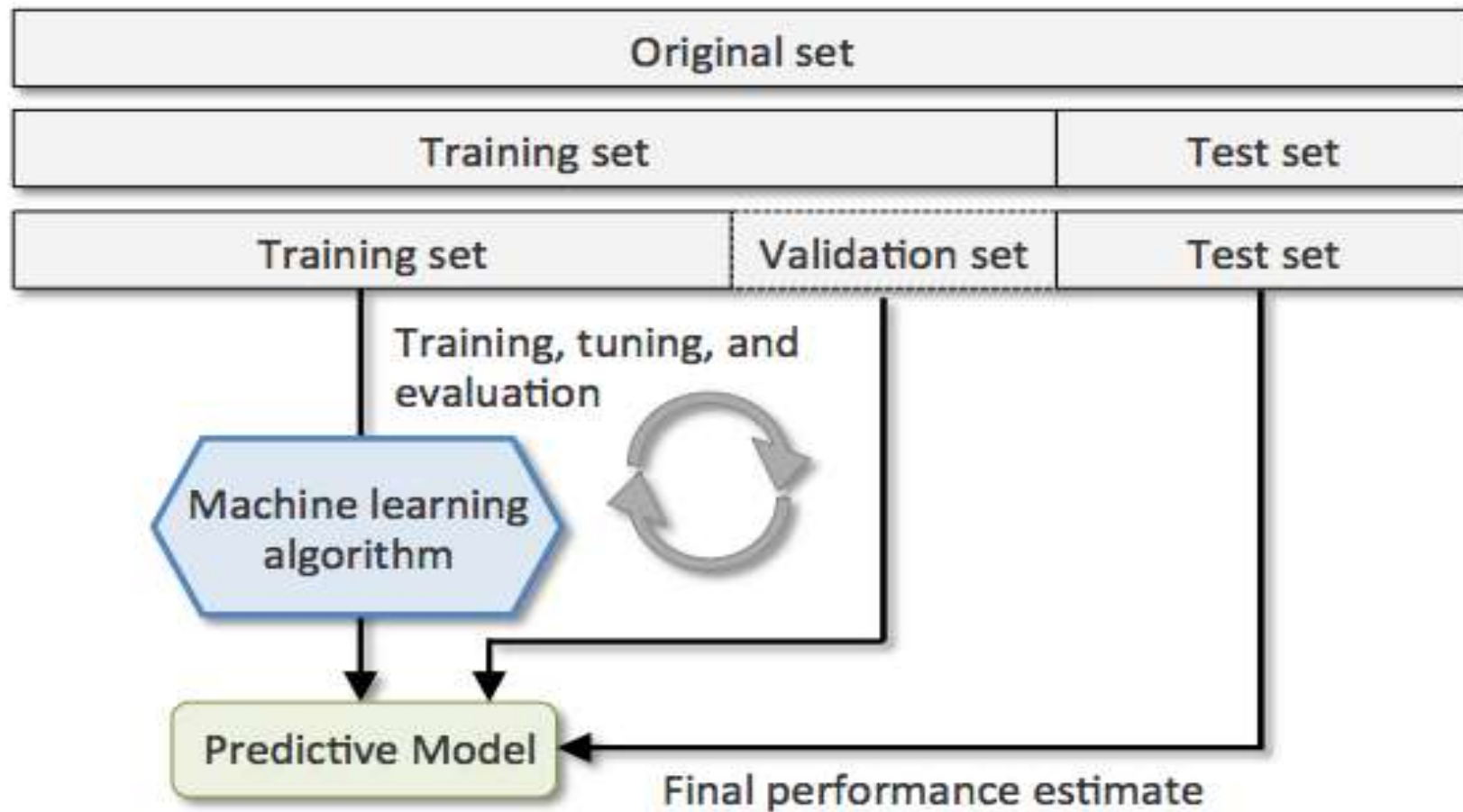


To make sure the models
are not overfitting



To determine the
accuracy of the models

Tùy theo bài toán chọn tỉ lệ phù
hợp giữa các tập dữ liệu



Performance Metrics

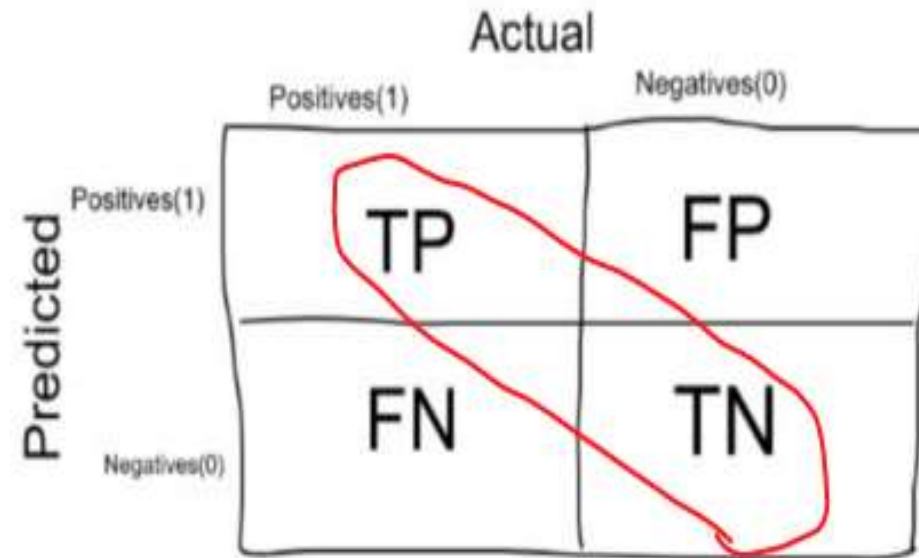
- **Confusion Matrix:**
- **True Positives (TP):** True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)
- **True Negatives (TN):** True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False)
- **False Positives (FP):** False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True).
- **False Negatives (FN):** False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False).

		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

Performance Metrics

- **Accuracy:** Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.

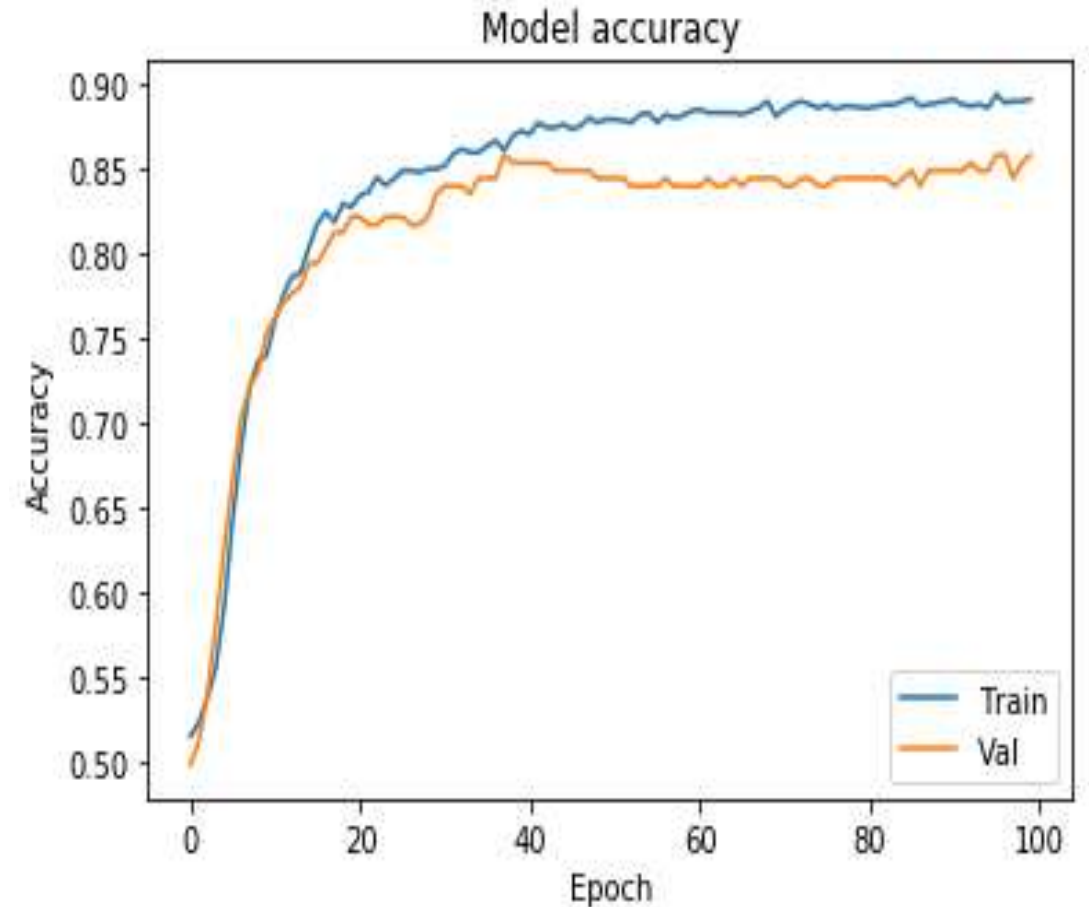
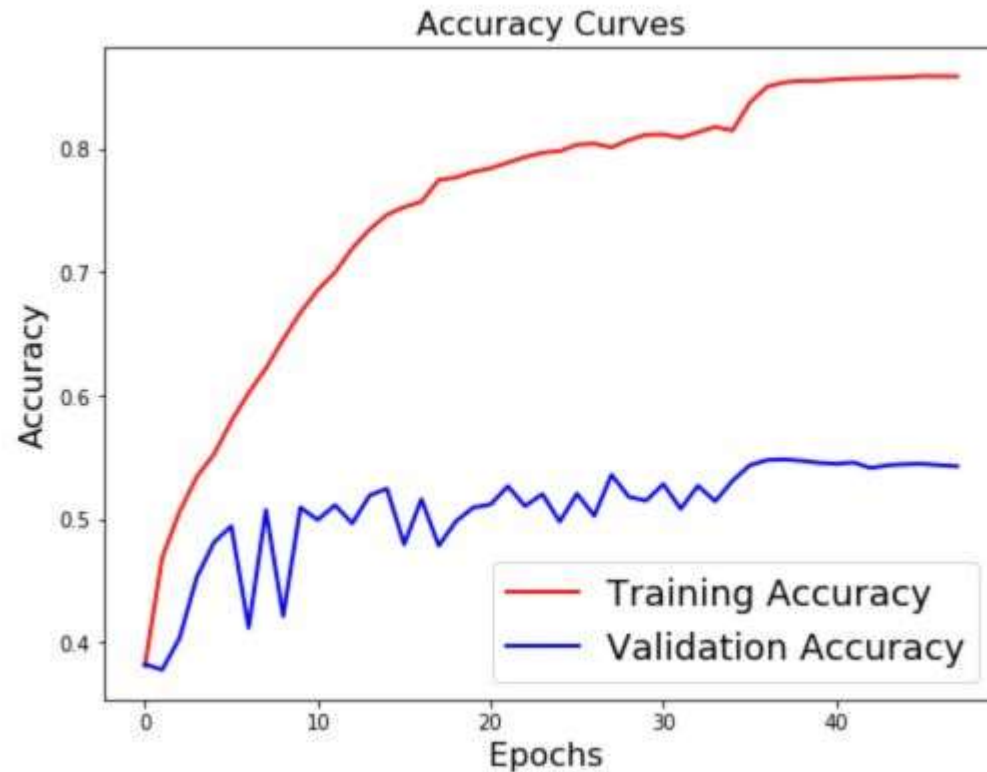
		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN



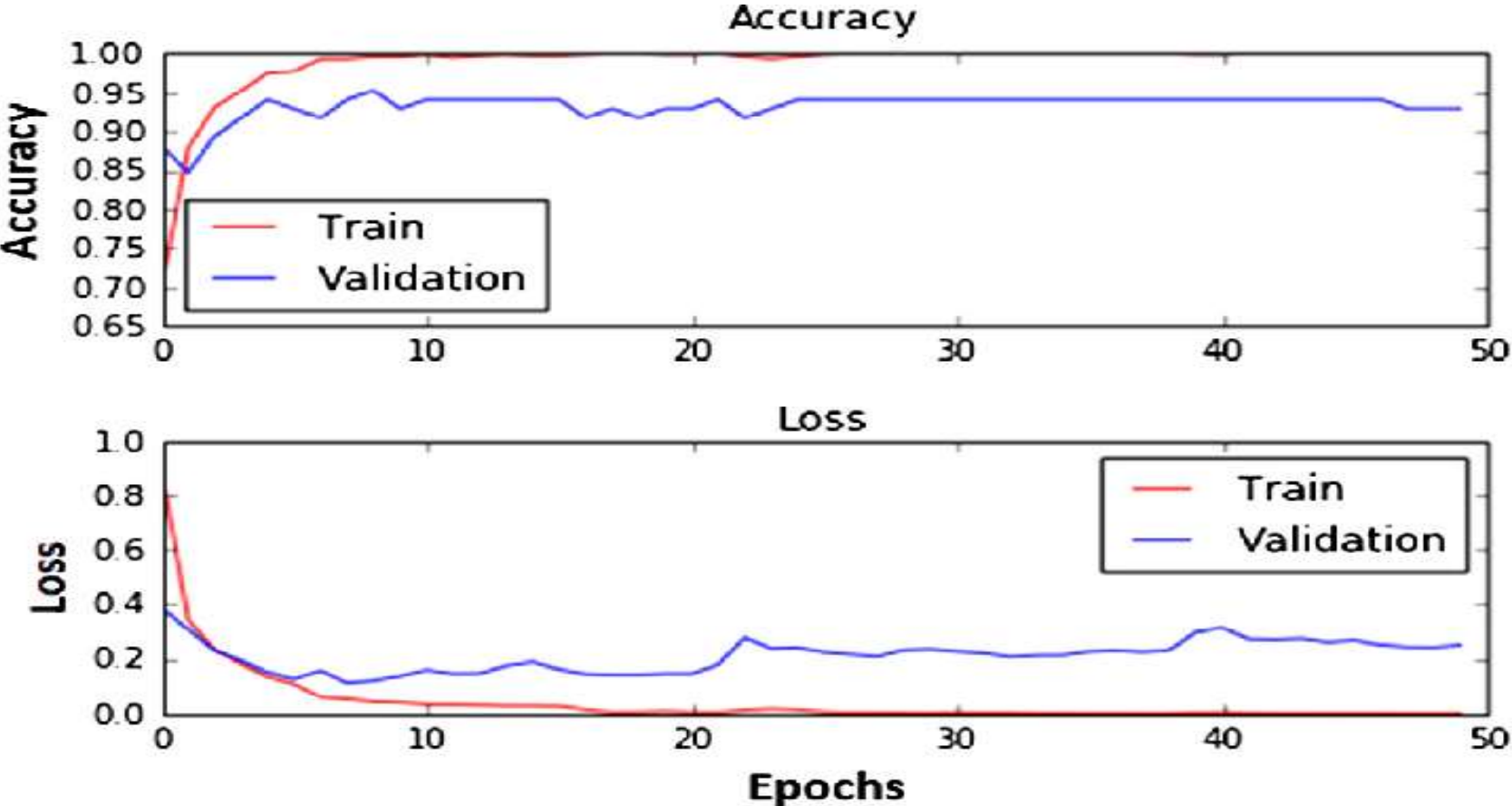
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Training and validation accuracy

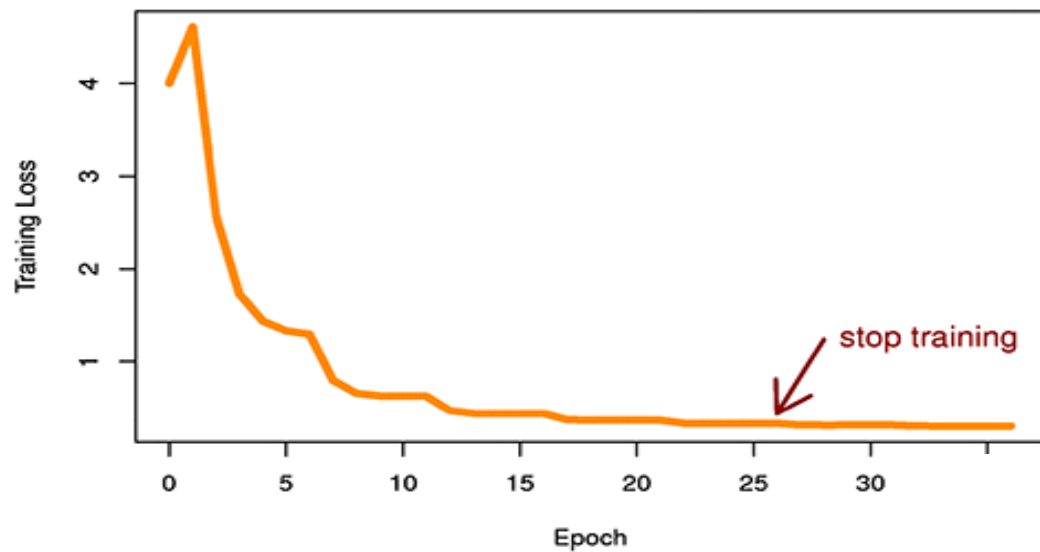
- **Overfitting**



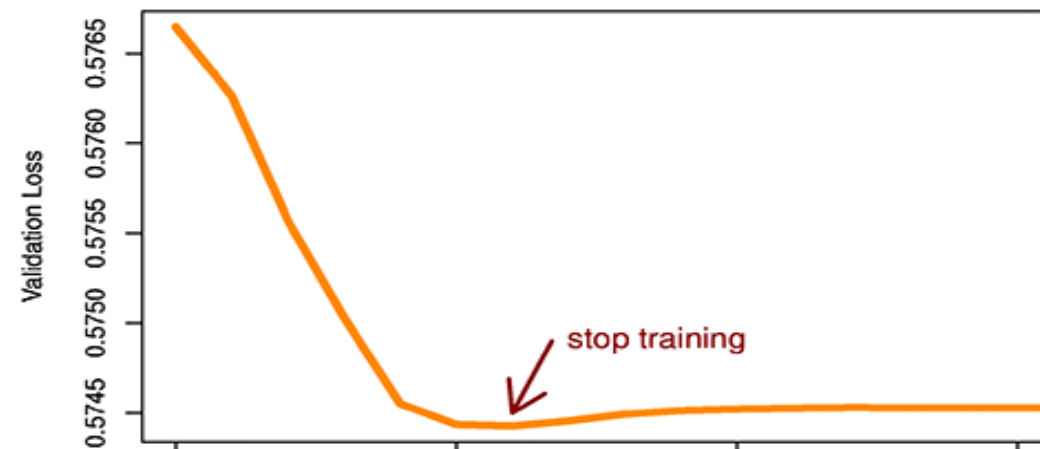
Training loss, Validation loss



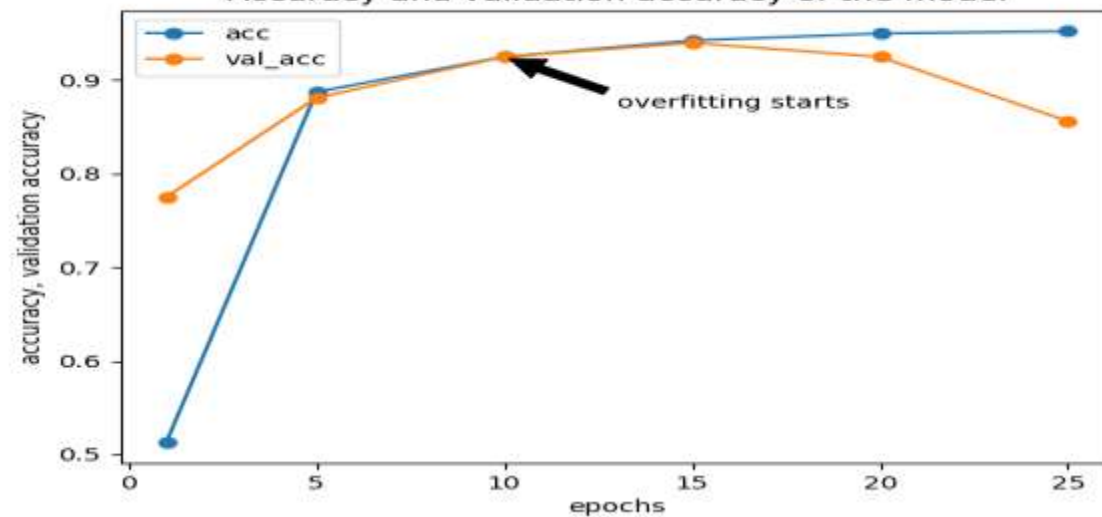
Typical Training Loss Curve



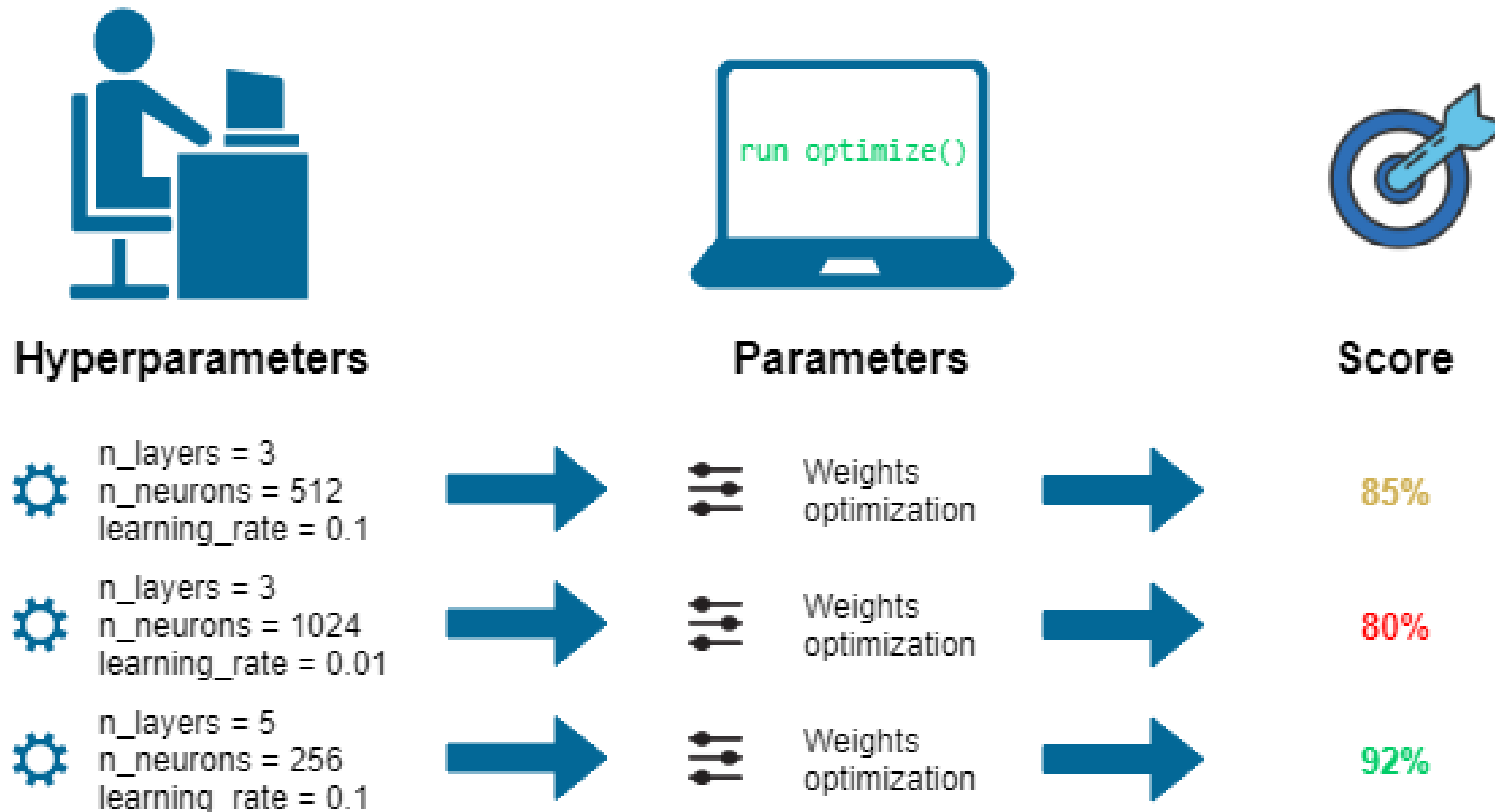
Typical Validation Loss Curve



Accuracy and validation accuracy of the model



Hyperparameters Optimization



- ***Model parameters:*** Weights and Biases
- ***Model Hyperparameters :***
 - Learning Rate
 - Number of Epochs
 - Hidden Layers
 - Hidden Units
 - Activations Functions
 - Dropout
 - Batch size

Hyperparameters Optimization

- model gives high accuracy on the training set (sample data) but fails to achieve good accuracy on the test set.
- increase the generalization capacity of the neural network, the model should be trained for an optimal number of epochs.



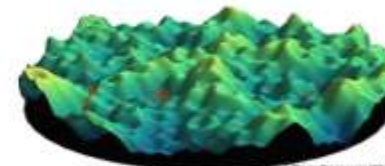
Data



Act/Grad/Filter



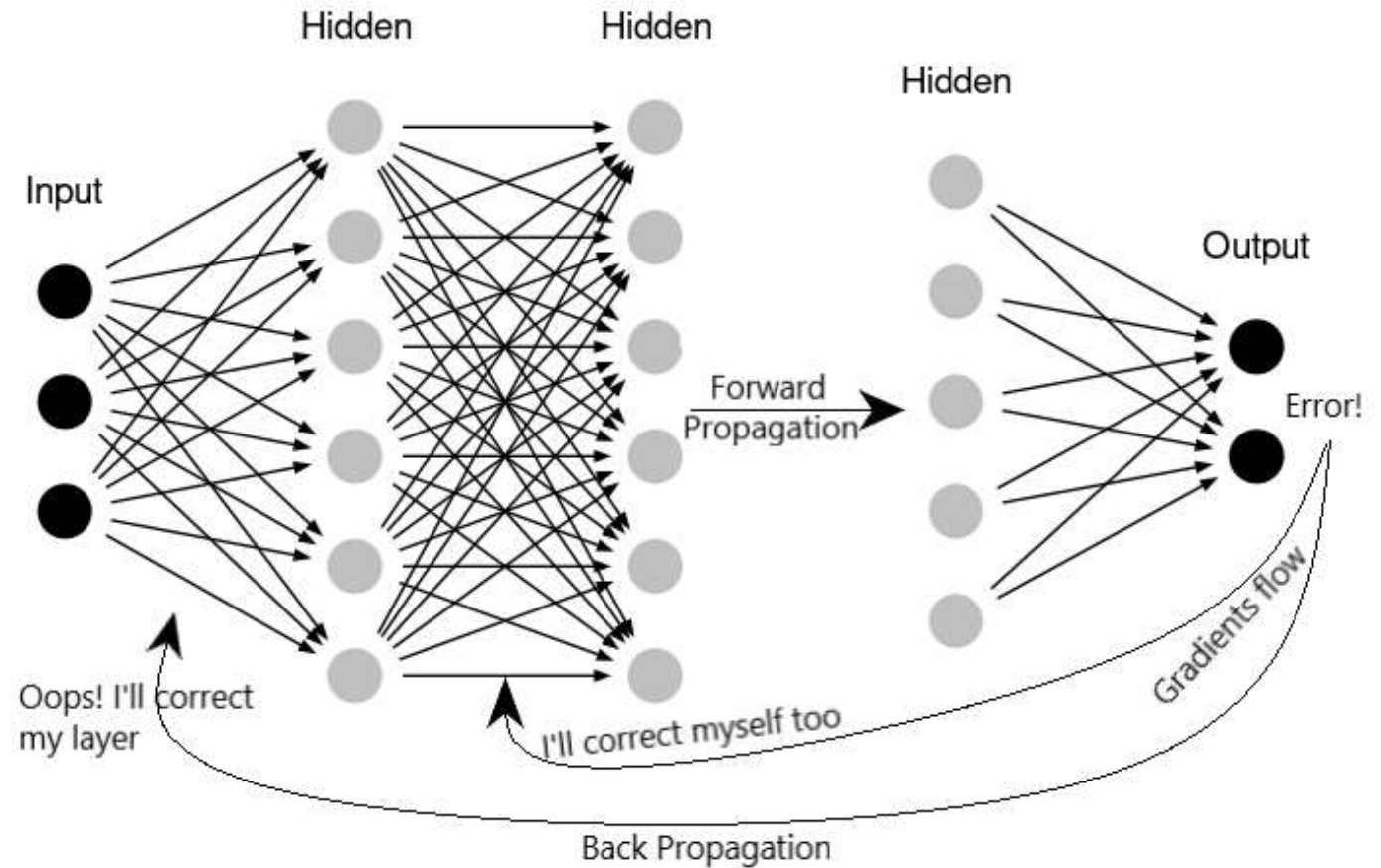
Metric



Space

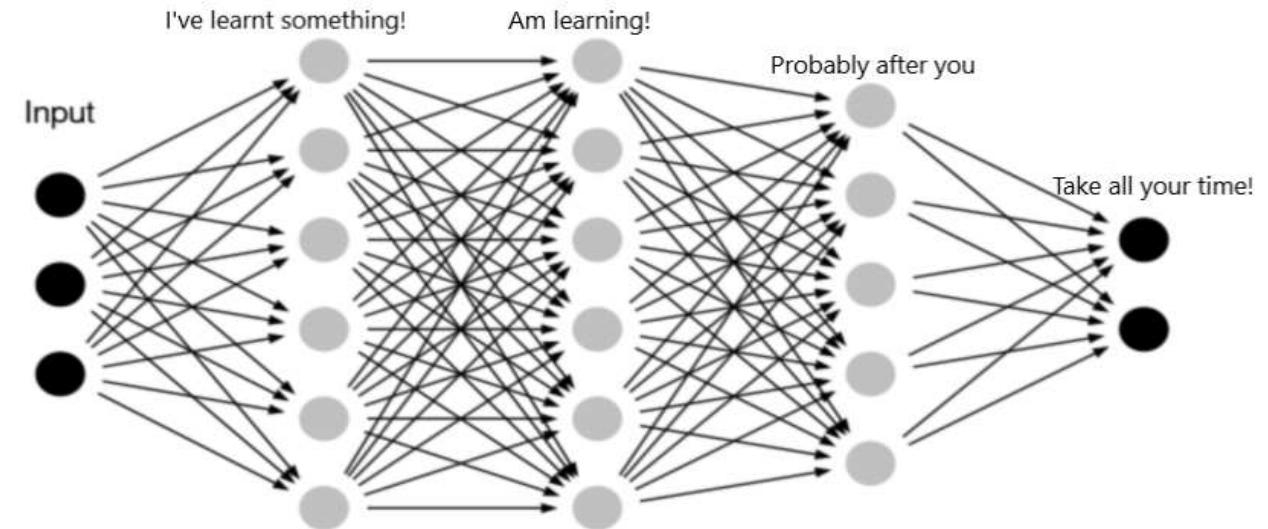
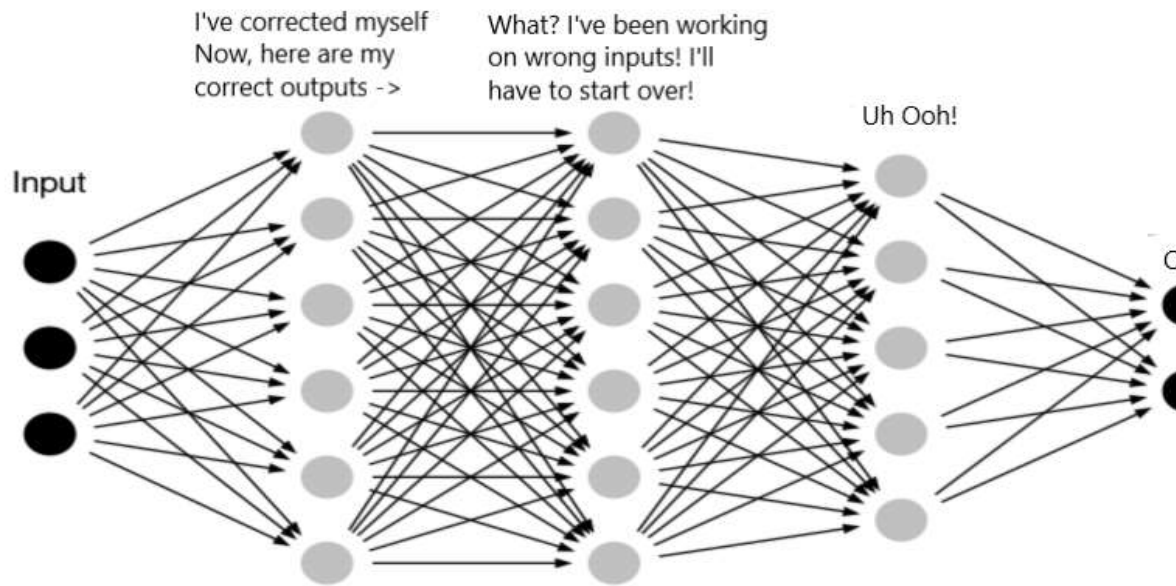
Batch Normalization

Neural networks learn the problem using BackPropagation algorithm to correct their “weights” and “biases”

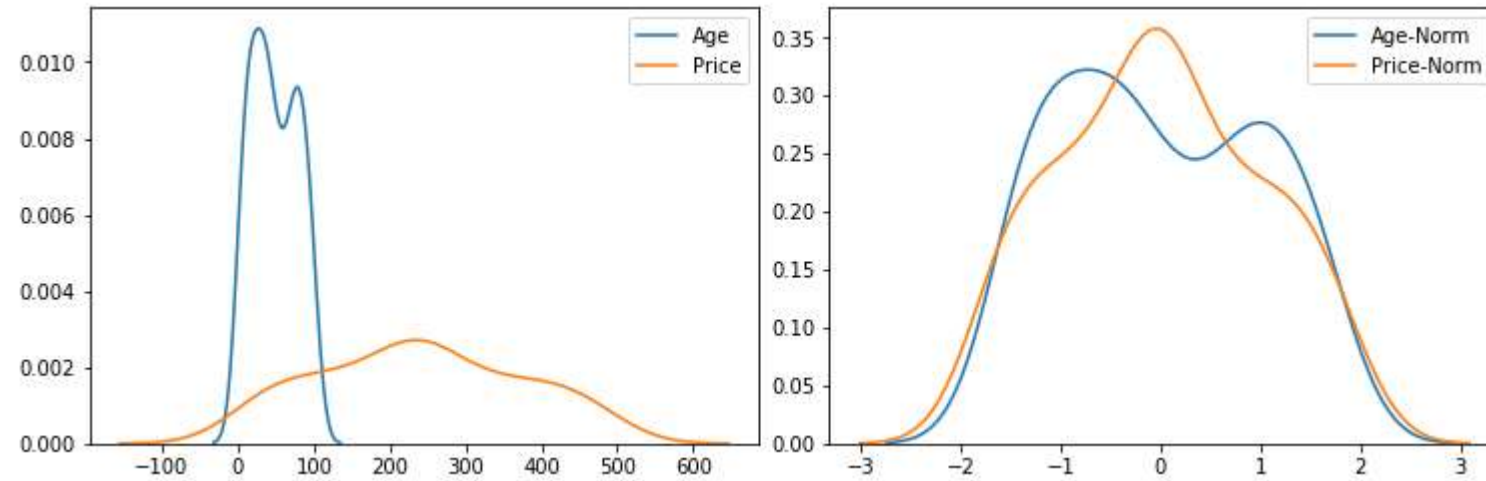


Batch Normalization

Internal Covariate Shift



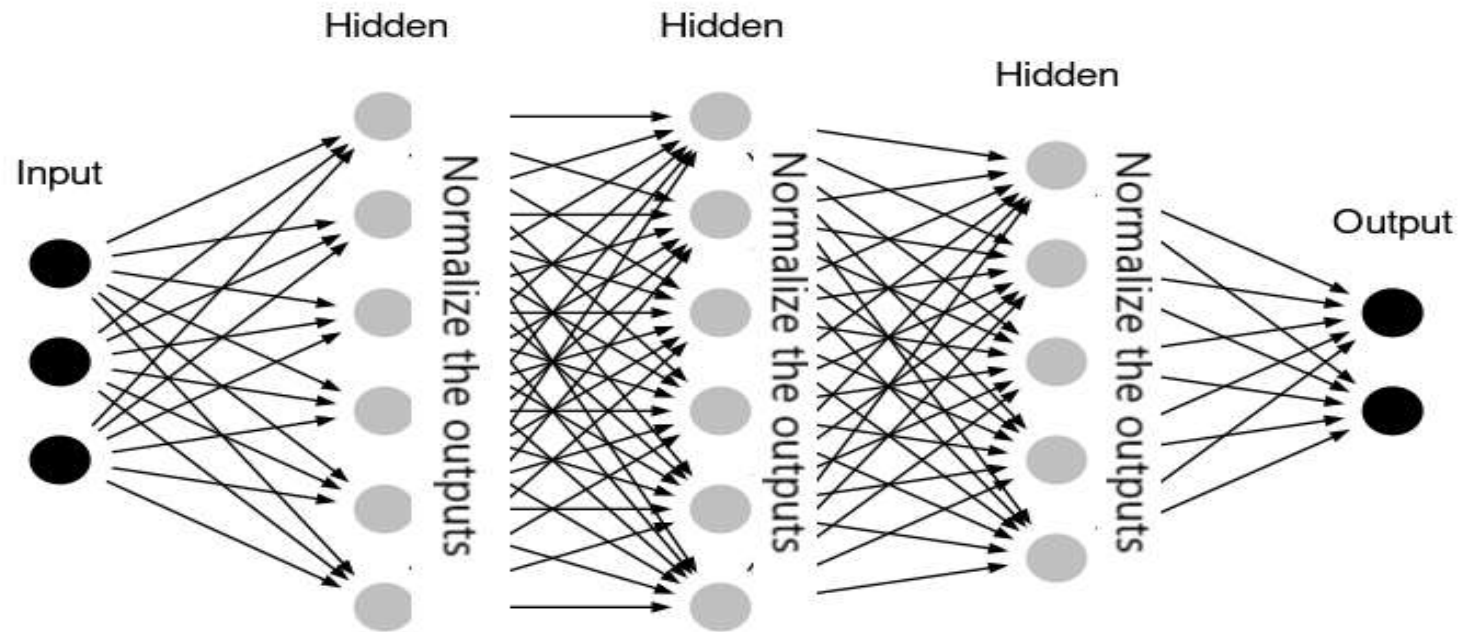
Normalization



Normalization is to convert the distribution of all inputs to have **mean=0** and **standard deviation=1**. So most of the values lie between -1 and 1.

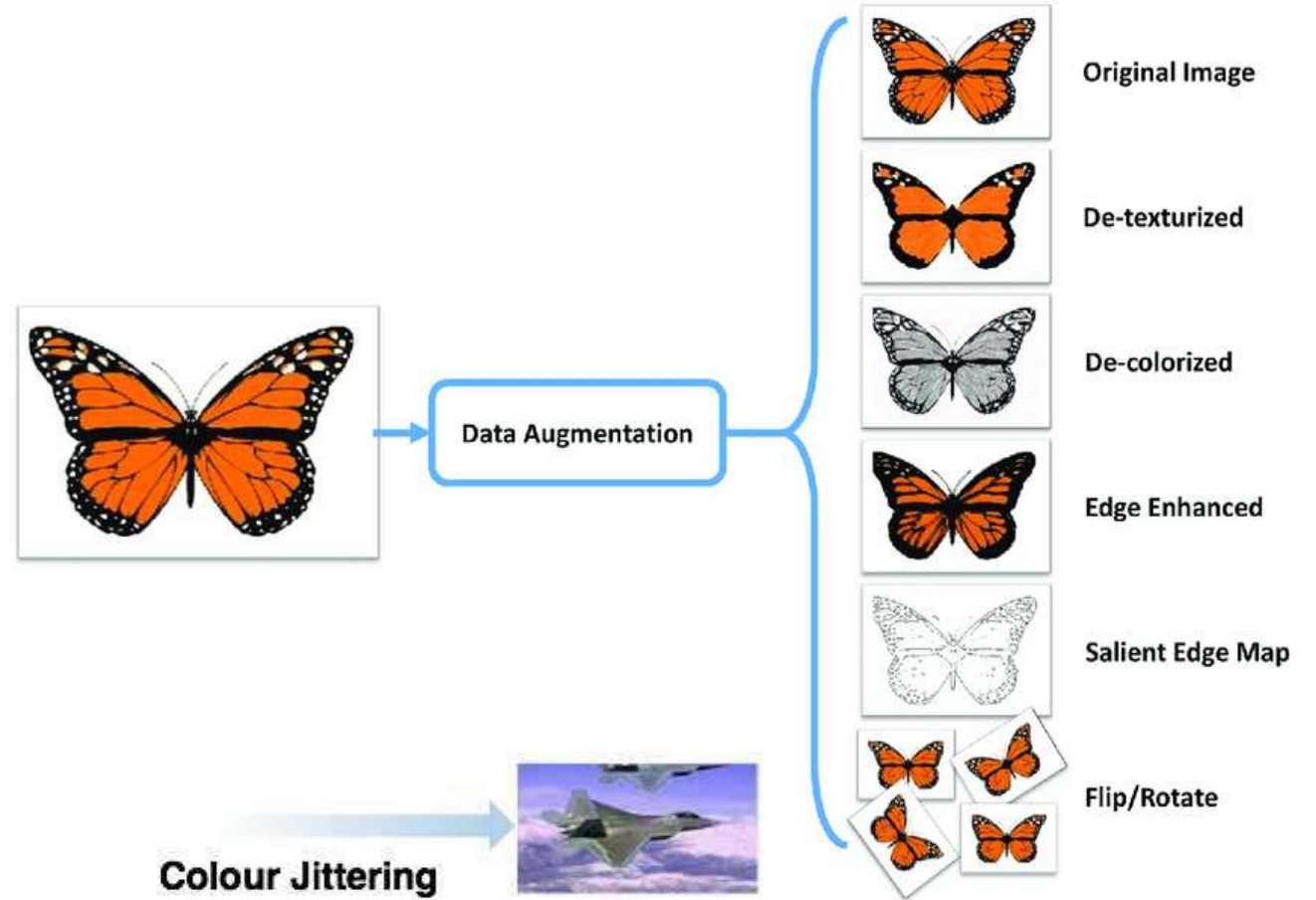
Batch Normalization





- To reduce this problem of internal covariate shift, Batch Normalization adds Normalization “layer” between each layers.
- normalization has to be done separately for each dimension (input neuron), over the ‘mini-batches’= > Batch normalization







Data augmentation

- Increase the diversity of data available for training models, without actually collecting new data.
- Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks



Original	Flip	Rotation	Random crop
			
<ul style="list-style-type: none"> • Image without any modification 	<ul style="list-style-type: none"> • Flipped with respect to an axis for which the meaning of the image is preserved 	<ul style="list-style-type: none"> • Rotation with a slight angle • Simulates incorrect horizon calibration 	<ul style="list-style-type: none"> • Random focus on one part of the image • Several random crops can be done in a row

Color shift	Noise addition	Information loss	Contrast change
			
<ul style="list-style-type: none"> • Nuances of RGB is slightly changed • Captures noise that can occur with light exposure 	<ul style="list-style-type: none"> • Addition of noise • More tolerance to quality variation of inputs 	<ul style="list-style-type: none"> • Parts of image ignored • Mimics potential loss of parts of image 	<ul style="list-style-type: none"> • Luminosity changes • Controls difference in exposition due to time of day

- Dùng GAN



winter Yosemite → summer Yosemite



summer Yosemite → winter Yosemite



Original photo

Reference photo

Result

Vấn đề của data augmentation

- Tính phụ thuộc dữ liệu và ứng dụng
- Sự đa dạng của augmentation
- Cách augmentation tốt nhất?
 - Auto augmentation
 - Cách Augmentation có thể transfer được !!!

Data augmentation in Keras

- provides the [ImageDataGenerator](#) class that defines the configuration for image data
 - Sample-wise standardization.
 - Feature-wise standardization.
 - ZCA whitening.
 - Random rotation, shifts, shear and flips.
 - Dimension reordering.
 - Save augmented images to disk.

```
datagen = ImageDataGenerator( )
```

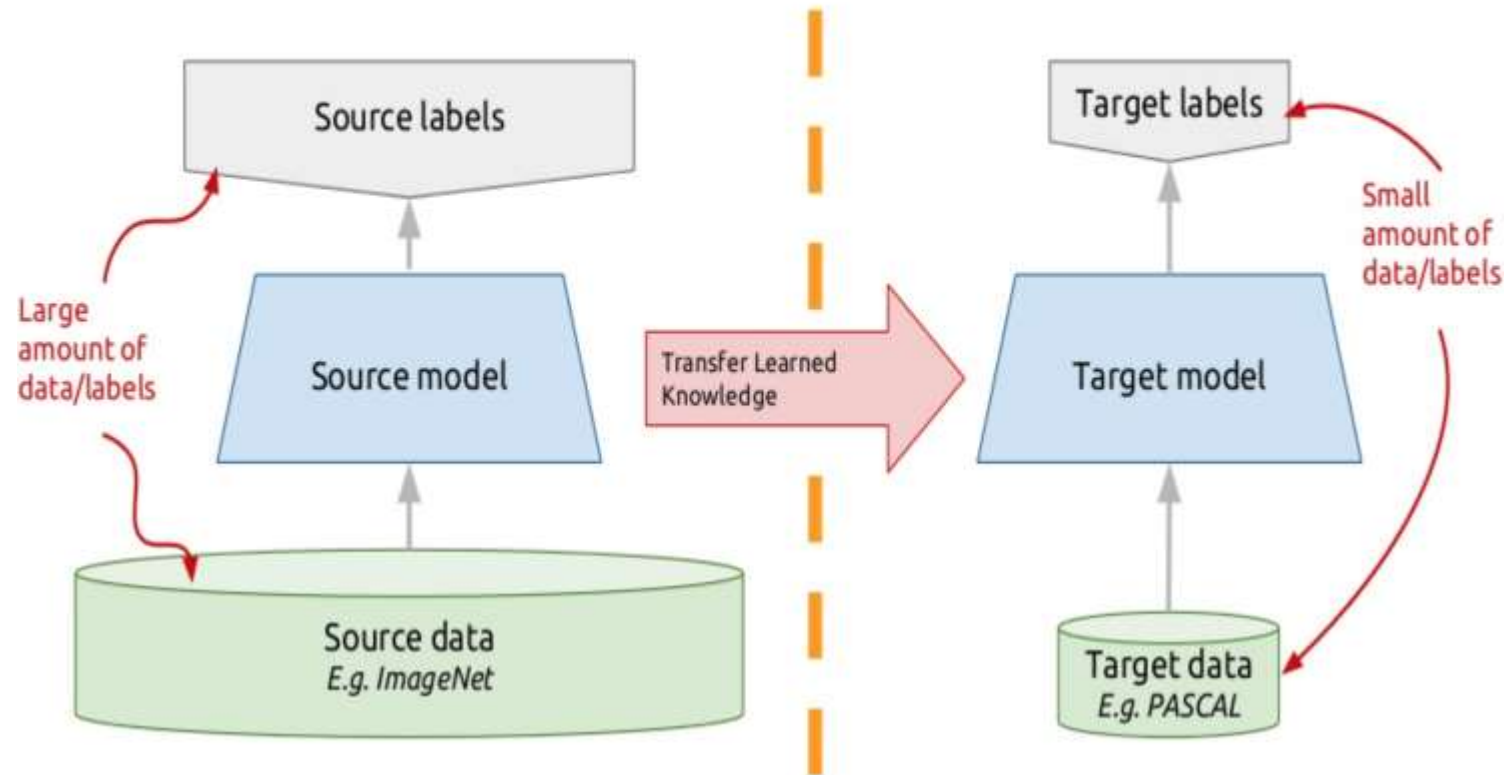
Data augmentation in Keras

- *datagen.fit(train)* : After configuring your **ImageDataGenerator**, pass it your training dataset.
- The data generator itself is in fact an iterator, returning batches of image samples when requested. the **flow()** function:
- *X_batch, y_batch = datagen.flow(train, train, batch_size=32)*

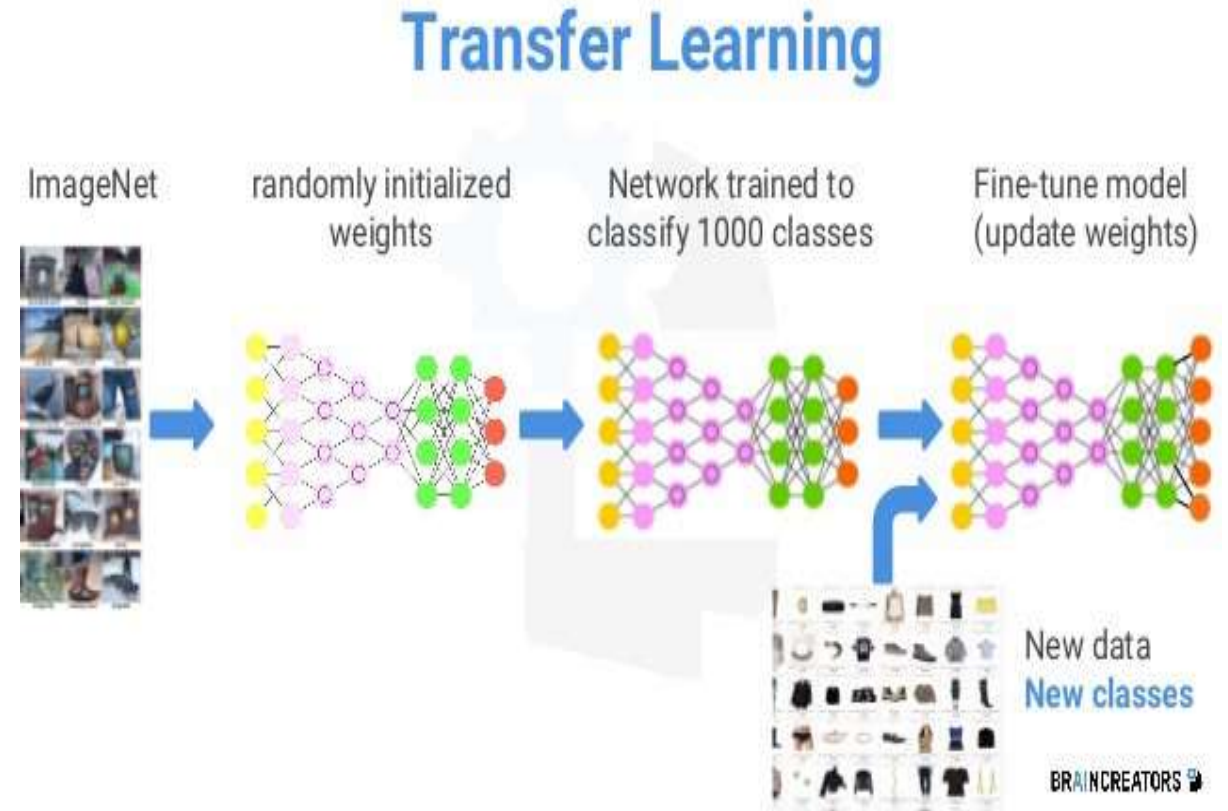
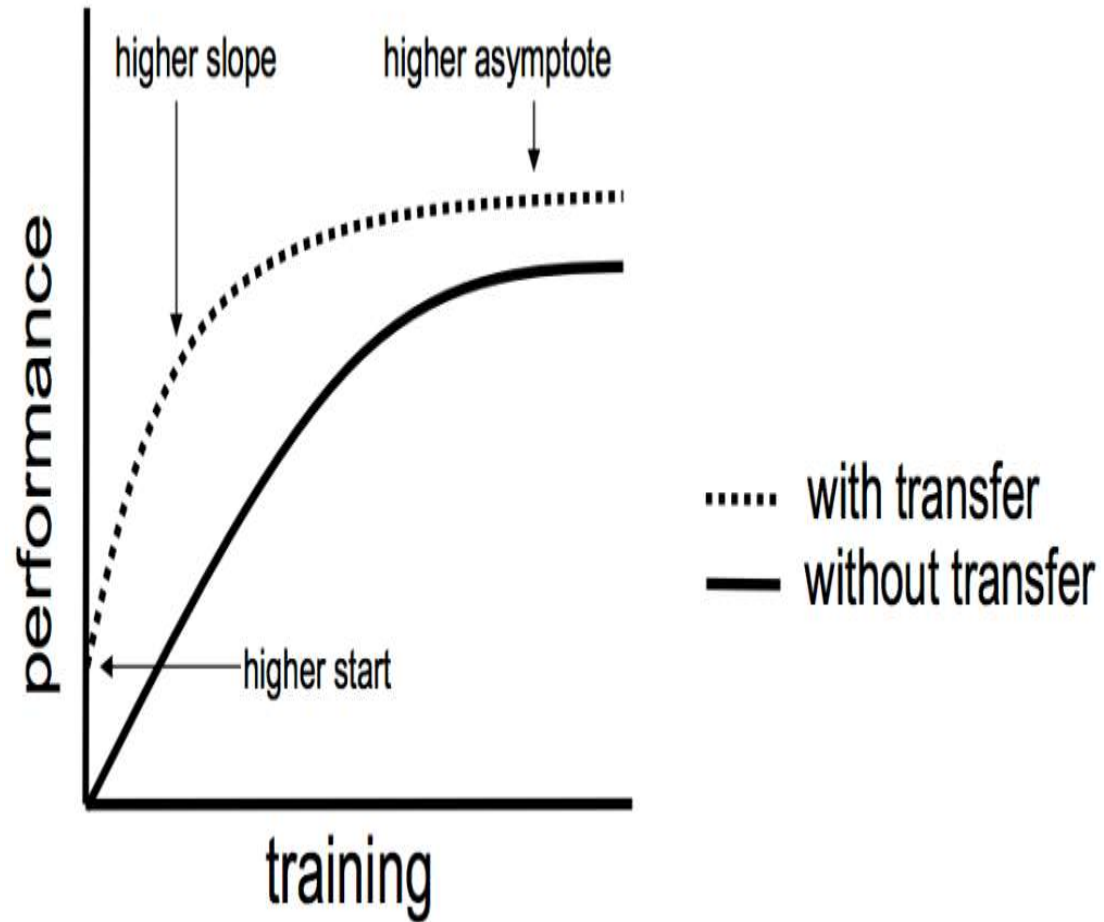
Transfer learning

- nhận diện 1000 người, dữ liệu để train chỉ khoảng 10 ảnh / 1 người
- => Số lượng dữ liệu là quá ít để train một mô hình CNN hoàn chỉnh.
- [VGGFace2 dataset](#) có 3.31 triệu ảnh của 9131 người, với trung bình 362.6 ảnh cho mỗi người. => đã xây dựng CNN model với accuracy hơn 99%.
- convolutional layer có tác dụng lấy ra các đặc trưng của ảnh
- convolutional layer + pooling layer (ConvNet) thì **model sẽ học được các đặc điểm của ảnh**, trước khi được cho vào fully connected layer => ConvNet trong VGGFace2 model cũng lấy ra được các đặc điểm của mặt người (tai, mũi, tóc,...)
- => có thể áp dụng phần ConvNet của VGGFace2 model vào bài toán nhận diện mặt người

Transfer learning: idea



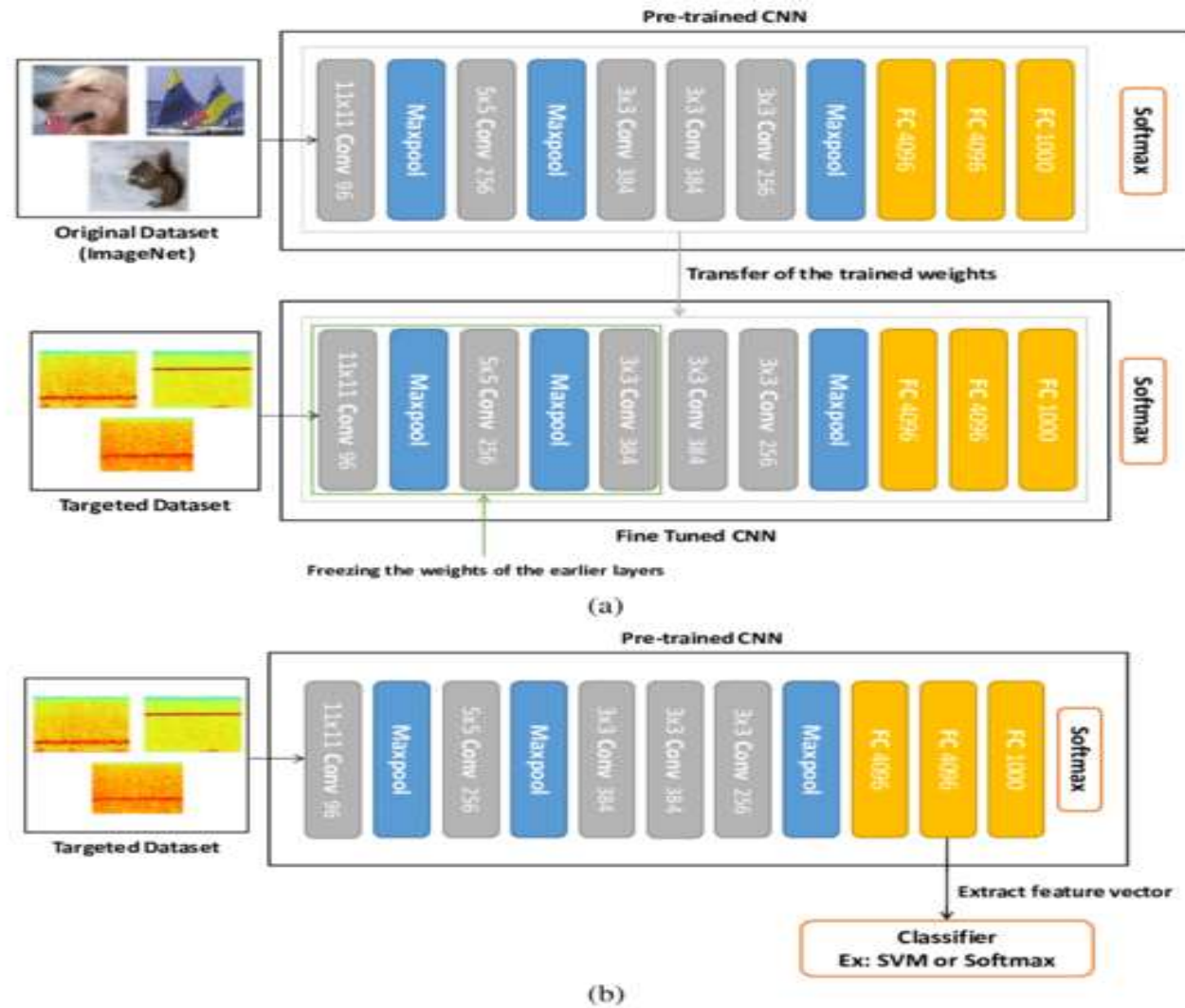
Transfer learning



Transfer learning

- Có 2 loại transfer learning:
- **Feature extractor:**
- Sau khi lấy ra các đặc điểm của ảnh bằng việc sử dụng ConvNet của pre-trained model, linear classifier (linear [SVM](#), softmax classifier,..) để phân loại ảnh.
- **Fine tuning:** Sau khi lấy ra các đặc điểm của ảnh bằng việc sử dụng ConvNet của pre-trained model, thì ta sẽ coi đây là input của 1 CNN mới bằng cách thêm các ConvNet và Fully Connected layer. Lý do là ConvNet của VGGFace 2 model có thể lấy ra được các thuộc tính của mặt người nói chung nhưng người Việt Nam có những đặc tính khác nên cần thêm 1 số Convnet mới để học thêm các thuộc tính của người Việt Nam.

Transfer learning



Pre-trained CNN model

- **VGG-16:** VGG được training trên tập dữ liệu của ImageNet có 1000 class nên ở Fully-connected layer cuối cùng sẽ có 1000 units.
- `from keras.applications.vgg16 import VGG16`
- Sử dụng pre-trained weight từ ImageNet và không sử dụng các Fully-connected layer ở cuối:

Feature extractor

- giữ lại phần ConvNet trong CNN và bỏ đi FCs
- output của ConvNet còn lại để làm input cho Logistic Regression với nhiều output,

Fine tuning

- giữ lại phần ConvNet trong CNN và bỏ đi FCs.

Computer vision task

R-CNN → **OverFeat** → MultiBox → SPP-Net → MR-CNN → DeepBox → AttentionNet →
2013.11 ICLR' 14 CVPR' 14 ECCV' 14 ICCV' 15 ICCV' 15 ICCV' 15

Fast R-CNN → DeepProposal → **Faster R-CNN** → **OHEM** → **YOLO v1** → G-CNN → AZNet →
ICCV' 15 ICCV' 15 NIPS' 15 CVPR' 16 CVPR' 16 CVPR' 16 CVPR' 16

Inside-OutsideNet(ION) → HyperNet → CRAFT → MultiPathNet(MPN) → **SSD** → GBDNet →
CVPR' 16 CVPR' 16 CVPR' 16 BMVC' 16 ECCV' 16 ECCV' 16

CPF → MS-CNN → **R-FCN** → PVANET → DeepID-Net → NoC → DSSD → TDM → **YOLO v2** →
ECCV' 16 ECCV' 16 NIPS' 16 NIPSW' 16 PAMI' 16 TPAMI' 16 arXiv' 17 CVPR' 17 CVPR' 17

Feature Pyramid Net(**FPN**) → RON → DCN → DeNet → CoupleNet → **RetinaNet** → DSOD →
CVPR' 17 CVPR' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17

Mask R-CNN → SMN → **YOLO v3** → SIN → STDN → **RefineDet** → MLKP → Relation-Net →
ICCV' 17 ICCV' 17 arXiv' 18 CVPR' 18 CVPR' 18 CVPR' 18 CVPR' 18 CVPR' 18

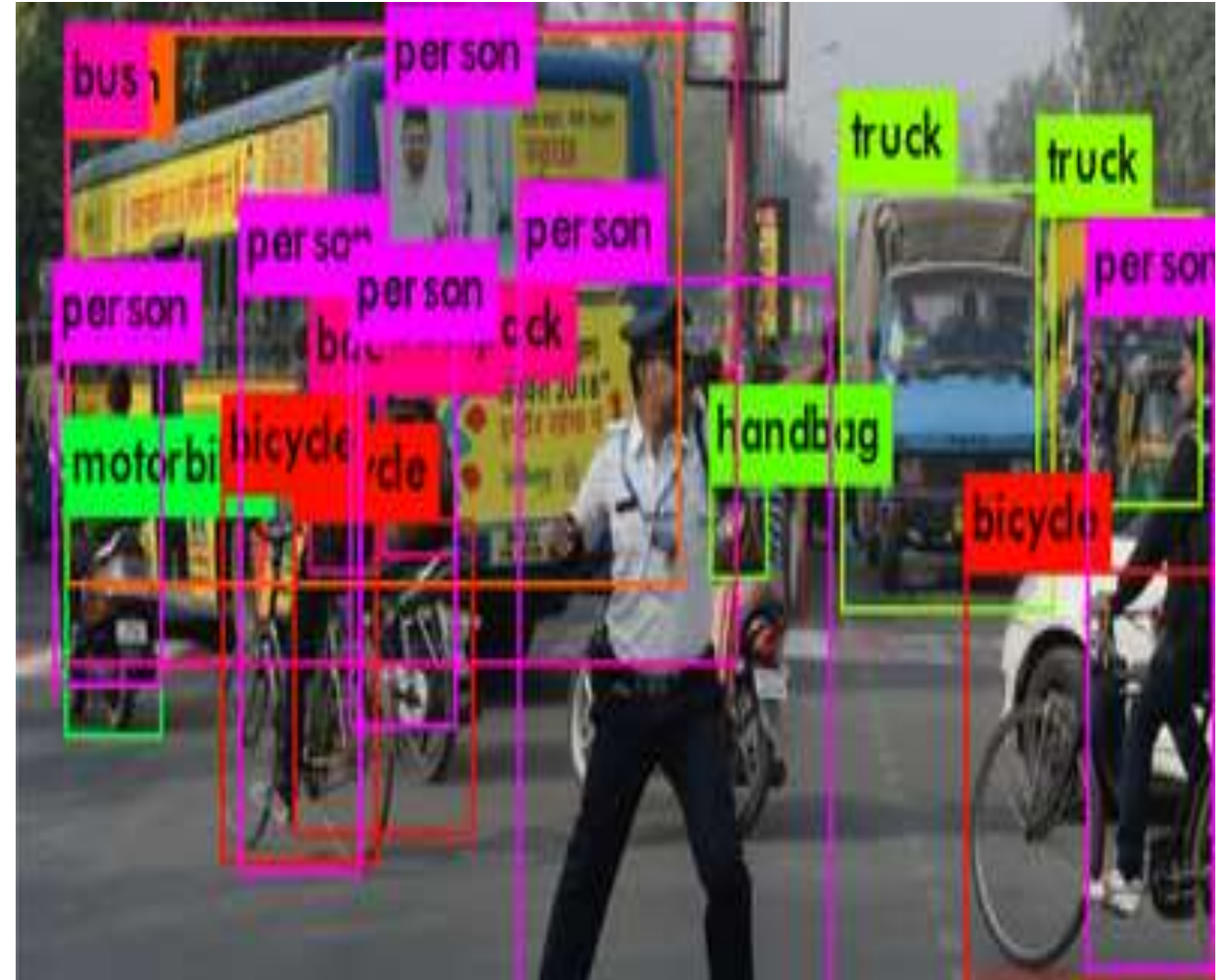
Cascade R-CNN → RFBNet → CornerNet → PFPNet → Pelee → HKRM → R-DAD → **M2Det** ...
CVPR' 18 ECCV' 18 ECCV' 18 ECCV' 18 NIPS' 18 NIPS' 18 AAAI' 19 AAAI' 19

Bài toán object detection

Xác định các bounding box (hình chữ nhật) quanh đối tượng

Với mỗi bounding box thì cần phân loại xem đây là đối tượng gì (chó, ngựa, ô tô,...) với bao nhiêu phần trăm chắc chắn.

- ⇒ có bao nhiêu đối tượng trong ảnh ? thiết kế được output layer hiệu quả hay không?
- ⇒ R-CNN (regional convolutional neural network)
- ⇒ Faster **Faster R-CNN**



- Ý tưởng thuật toán R-CNN khá đơn giản:
- Bước 1: Dùng Selective Search algorithm để lấy ra khoảng 2000 bounding box trong input
- mà có khả năng chứa đối tượng.
- Bước 2: Với mỗi bounding box ta xác định xem nó là đối tượng nào (người, ô tô, xe đạp,...)