Quiz

- Maximum 20 minutes
- 10 minutes later → first check
- 15 minuets later → second check
- Nothing is allowed on the table except test papers and pens → 0 point
- Cheat behaviors → 0 point

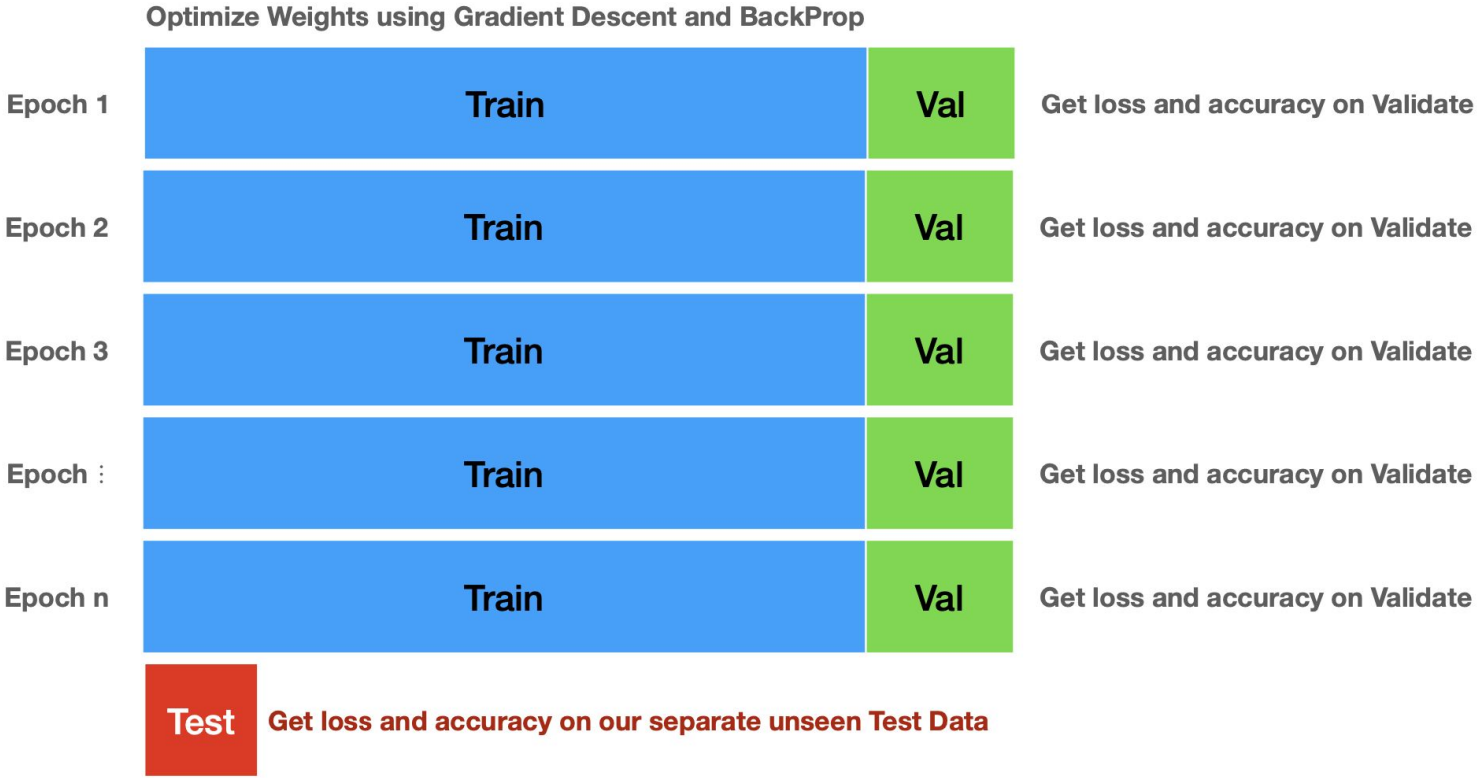
模型品質與改進

授課老師: 楊景明

評估模型

Training, Validation, and Test Datasets

- 模型使用**Training data**來學習參數
- **Validation data**是我們在訓練期間用來對模型進行公正評估的資料集（每個epoch訓練完出現）
- **Test data**通常是指我們最終測試模型效能的資料集（Ex: Kaggle competetion）
- ”**Validation**“和”**Test**“常常被交換使用

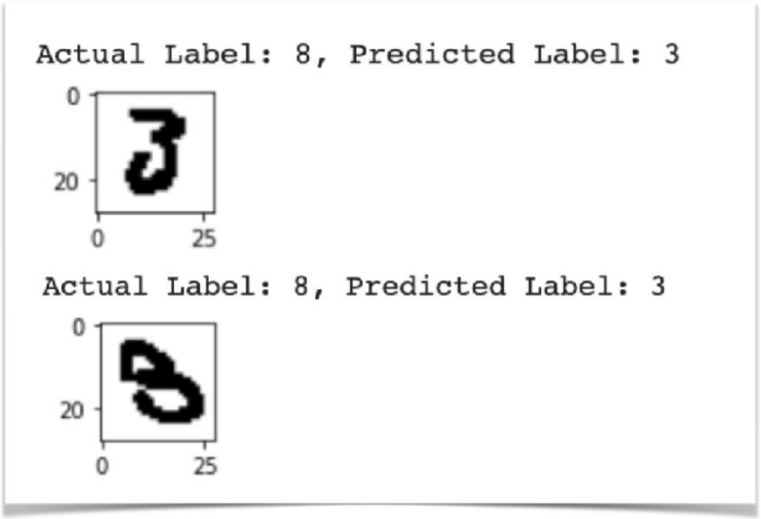

Optimize Weights using Gradient Descent and BackProp
Epoch 1
Train
Val
Get loss and accuracy on Validate
Epoch 2
Train
Val
Get loss and accuracy on Validate
Epoch 3
Train
Val
Get loss and accuracy on Validate
Epoch ⋮
Train
Val
Get loss and accuracy on Validate
Epoch n
Train
Val
Get loss and accuracy on Validate
Test
Get loss and accuracy on our separate unseen Test Data

基本的評估指標

- Training Loss
- Training Accuracy
- Test/Validation Loss
- Test/Validation Accuracy

假如模型無法分辨”3”和”8”

- 如何找出這種問題？
- 如何知道哪個類別表現得比較差？
- 如何知道模型傾向將”8”分辨成”3”？



Accuracy is not enough

MNIST Confusion Matrix Example

Predicted Labels

	0	1	2	3	4	5	6	7	8	9	
[[	973	0	2	0	0	1	1	1	2	0]	0
[	0	1128	1	1	0	0	2	1	2	0]	1
[	2	2	1018	1	1	0	0	4	4	0]	2
[	0	0	0	1001	0	5	0	0	4	0]	3
[	0	0	2	0	971	0	0	0	0	9]	4
[	2	0	0	3	0	884	1	0	0	2]	5
[	6	2	1	0	2	5	939	0	3	0]	6
[	0	2	7	1	0	0	0	1014	2	2]	7
[	5	0	3	0	0	1	1	2	959	3]	8
[	2	2	0	3	5	3	0	5	3	986]]	9

True Labels

- 綠色:預測正確
- 紅色:預測錯誤

找出容易預測錯誤的類別

Predicted Labels

	0	1	2	3	4	5	6	7	8	9	
[[	973	0	2	0	0	1	1	1	2	0]	0
[	0	1128	1	1	0	0	2	1	2	0]	1
[	2	2	1018	1	1	0	0	4	4	0]	2
[	0	0	0	1001	0	5	0	0	4	0]	3
[	0	0	2	0	971	0	0	0	0	9]	4
[	2	0	0	3	0	884	1	0	0	2]	5
[	6	2	1	0	2	5	939	0	3	0]	6
[	0	2	7	1	0	0	0	1014	2	2]	7
[	5	0	3	0	0	1	1	2	959	3]	8
[	2	2	0	3	5	3	0	5	3	986]]	9

True Labels

- 4容易被預測為9
- 7容易被預測為2

二元分類

預測病患是否有COVID…

N = 145	Predicted NO	Predicted YES	
True Label NO	True Negatives: 40	False Positives: 5	Total True Negatives: 45
True Label YES	False Negatives: 10	True Positives: 90	Total True Positives: 100
	Predicted Negatives: 50	Predicted Positives: 95	

- **Accuracy** = (TP + TN) / *Total* = (40 + 90) / 145 = 130 / 145 = 0.897
- **Misclassification** or **Error Rate** = 1 - *Accuracy* = 1 - 0.897 = 0.103

二元分類

預測病患是否有COVID…

N = 145	Predicted NO	Predicted YES	
True Label NO	True Negatives: 40	False Positives: 5	Total True Negatives: 45
True Label YES	False Negatives: 10	True Positives: 90	Total True Positives: 100
	Predicted Negatives: 50	Predicted Positives: 95	

- **True Positive Rate** or **Sensitivity** or **Recall** = TP / TruePositiveLabels = 90 / 100 = 0.9
  - 患有COVID的病人中，我們預測出了多少
- **False Positive Rate** = FP / TrueNegativeLabels = 5 / 45 = 0.11
  - 沒有患COVID的病人中，我們誤判了多少患者有得 COVID

二元分類

預測病患是否有COVID…

N = 145	Predicted NO	Predicted YES	
True Label NO	True Negatives: 40	False Positives: 5	Total True Negatives: 45
True Label YES	False Negatives: 10	True Positives: 90	Total True Positives: 100
	Predicted Negatives: 50	Predicted Positives: 95	

- **True Negative Rate** or **Specificity** = TN / TrueNegativeLabels = 40 / 45 = 0.89
  - 沒有患COVID的病人中，我們正確預測出了多少
- **Precision** = TP / PredictedYes = 90 / 95 = 0/95
  - 我們預測有得COVID的病患，有多少是正確的

Precision, Recall, F1

- **Precision/Recall** tradeoff - 通常提**升Precision或Recall**會導致另一方下降
- 有時候我們可以接受犧牲一些**Precision**來提高**Recall** (看診)
- **F1** - **Precision**和**Recall**的調和平均數(harmonic mean)

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Classification Report

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.99	0.99	0.99	1135
2	0.98	0.99	0.99	1032
3	0.99	0.99	0.99	1010
4	0.99	0.99	0.99	982
5	0.98	0.99	0.99	892
6	0.99	0.98	0.99	958
7	0.99	0.99	0.99	1028
8	0.98	0.98	0.98	974
9	0.98	0.98	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

- Support代表各類別出現的次數

Implementation

Download notebook at:
https://github.com/albert831229/nchu-computer-vision/tree/main/113/day

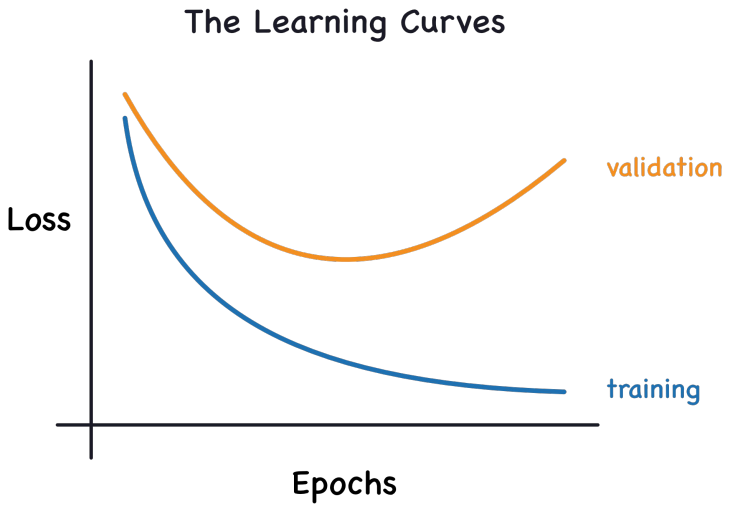
Overfitting and Generalization

如何避免Overfitting

- 如何讓模型取得好表現
  - 更多資料？
  - 更大更深的模型？
  - 更多Epochs?
- 在訓練期間，我們可以使用一些技巧來降低**Overfitting**以及增加**Generalization**

Overfitting and Generalisation

- 當模型在訓練資料上過度訓練，並在測試資料上表現很差 (很常發生)
- 當我們的訓練資料太少、使用太多Features、或使用太複雜的模型
- **Generalization**代表模型在沒見過的資料上的表現



Overfitting Example

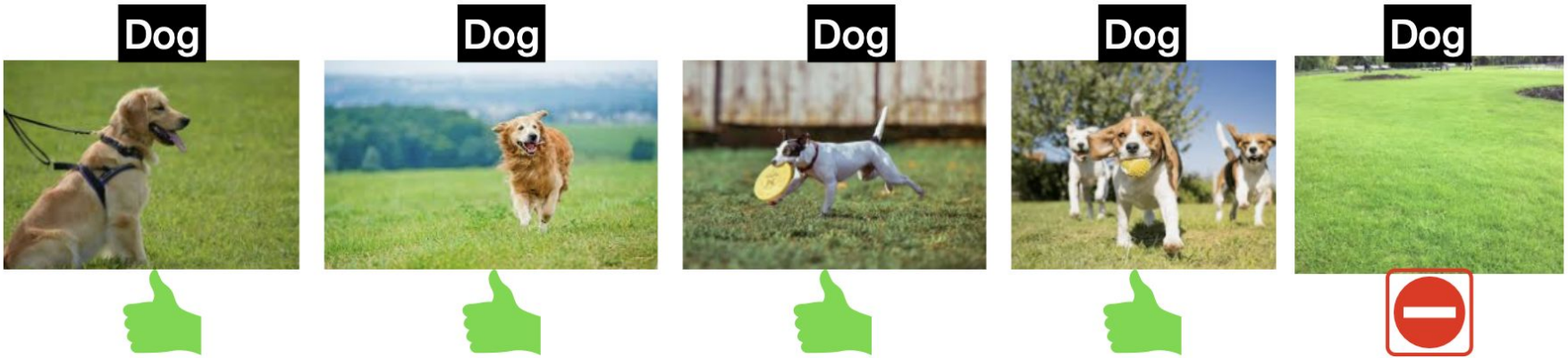



- 假設模型學會利用顏色分辨

- 消防車！ :(

如何避免Overfitting

- 最簡單方法：增加訓練資料或降低模型複雜度
- 使用**Regularization**：限制模型的複雜度，並讓模型透過正確的特徵學習
- Ex: 我們希望模型能夠學習和狗相關的特徵 (尾巴、鼻子、耳朵)，而不是其他一般的特徵 (樹、草)



如何欺騙模型

Source: https://twitter.com/drjuliashaw/status/874293864814845952

Regularization Methods

Regularization Methods

- L1 & L2 Regularization
- Drop Out
- Data Augmentation
- Early Stopping
- Batch Normalization

L1 and L2 Regularization

- 迫使模型使用較小的參數 (weights and biases)
- 避免特定的節點產生過大的影響

Loss Function + Penalty (L1 or L2)

L1 Regularization

- Also called L1-Norm or Lasso Regression

$$\text{Loss Function} + \lambda \sum_{j=1}^{p} |\beta_j|$$

- β_j: 參數
- λ: 控制Penalty的效果
- 越大的λ代表Penalty越重

L2 Regularization

- Also called Ridge Regression

$$LossFunction + \lambda \sum_{j=1}^{p} \beta_j^2$$

  - β_j: 參數
  - λ: 控制Penalty的效果
  - 越大的λ代表Penalty越重

L1與L2的區別

- L1會優先將不重要的特徵的權重降為0
  - 特徵選取
- L2會優先將所有的權重減小
  - 大的權重會產生更大的**Penalty**

$$\text{L1: } LossFunction + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$\text{L2: } LossFunction + \lambda \sum_{j=1}^{p} \beta_j^2$$

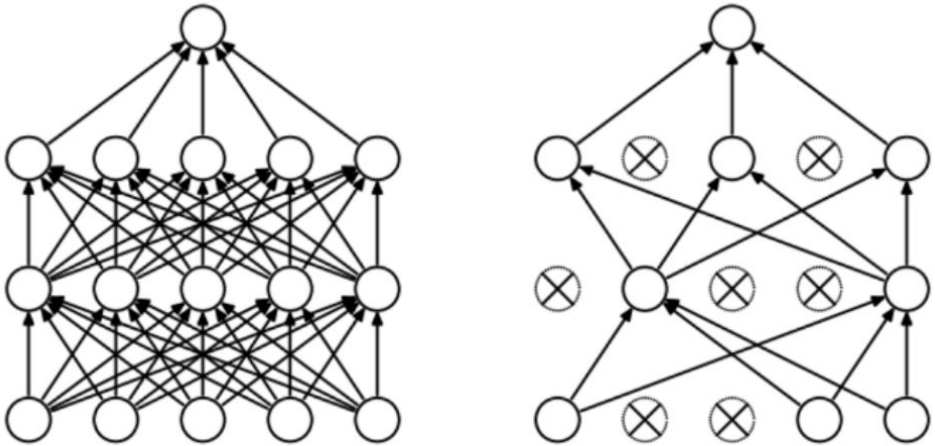
L1與L2總結

- L1和L2都會使模型使用較小的權重
- 除非使用大的權重可以顯著降低原有的Loss
- 在較小的權重和較低的原有Loss間做選擇 (透過λ控制)

Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

Dropout

- Dropout影響的不是訓練損失，而是模型本身
- 在訓練過程中，隨機關閉某一層的一些節點



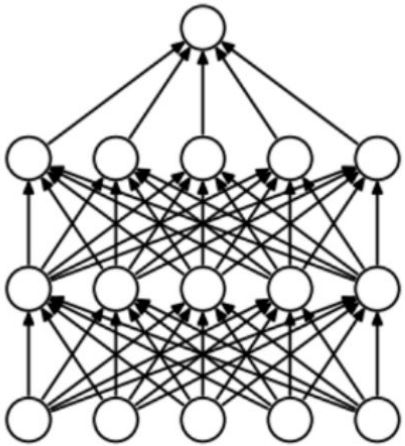
(a) Standard Neural Net

(b) After applying dropout.

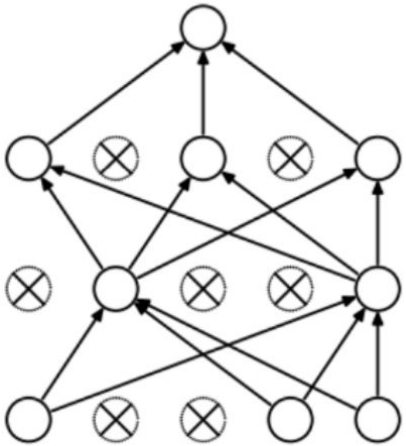
- 隨機關閉一些節點 (ex: 一半)
- 將一個Mini-Batch前向傳遞，並用反向傳遞計算梯度
- 更新開放的節點的權重
- 恢復所有節點、再次隨機關閉一些節點、並傳遞下一組Mini-Batch

Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

Dropout



(a) Standard Neural Net

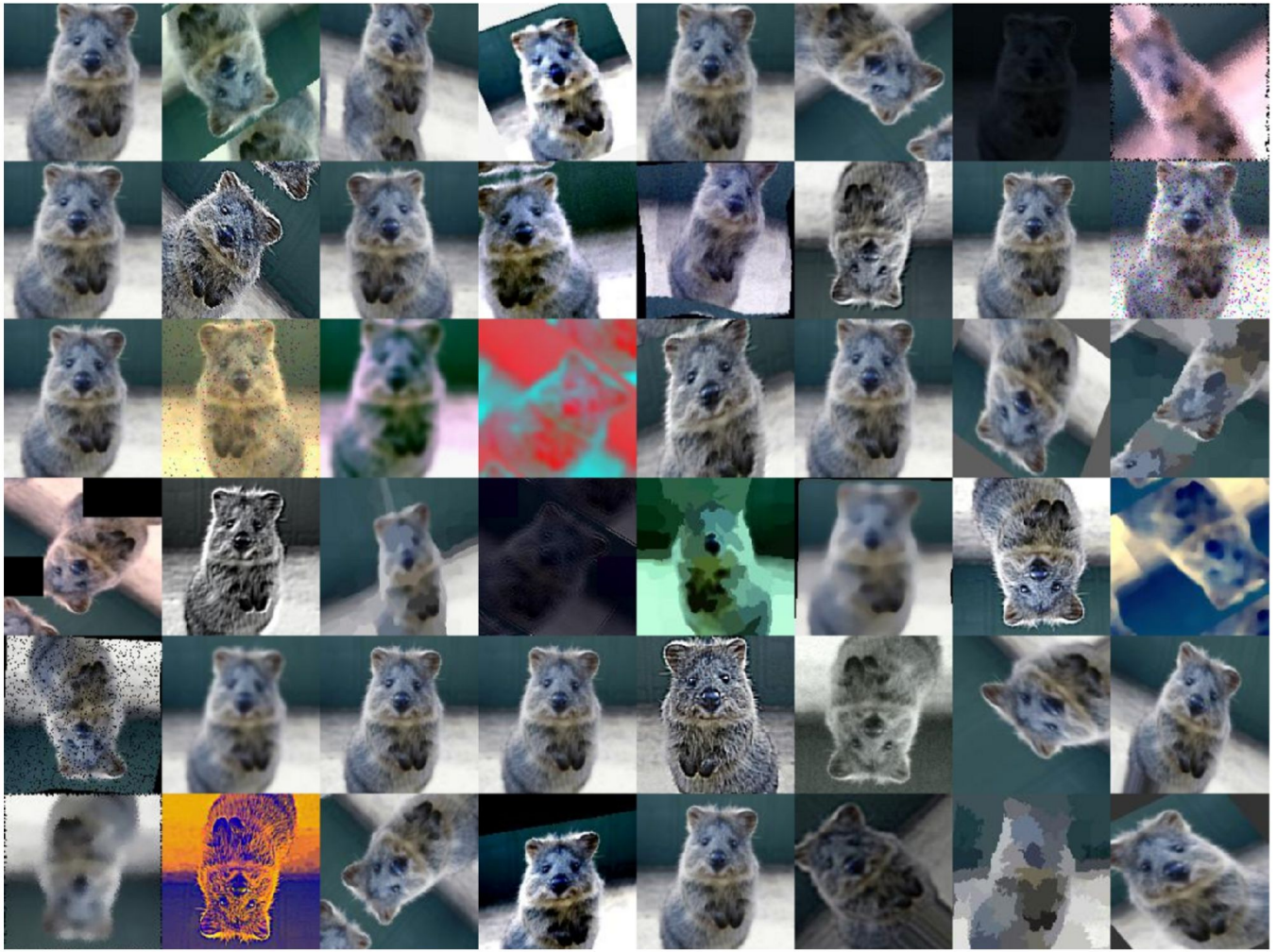


(b) After applying dropout.

- 使用Dropout Rate來控制關閉節點的數量
- 迫使模型學習更可靠的特徵(每次訓練不一樣的模型)
- Dropout也會增加模型Converge所需要的時間
- 在測試模型時，我們會使用所有的節點來做預測

Data Augmentation

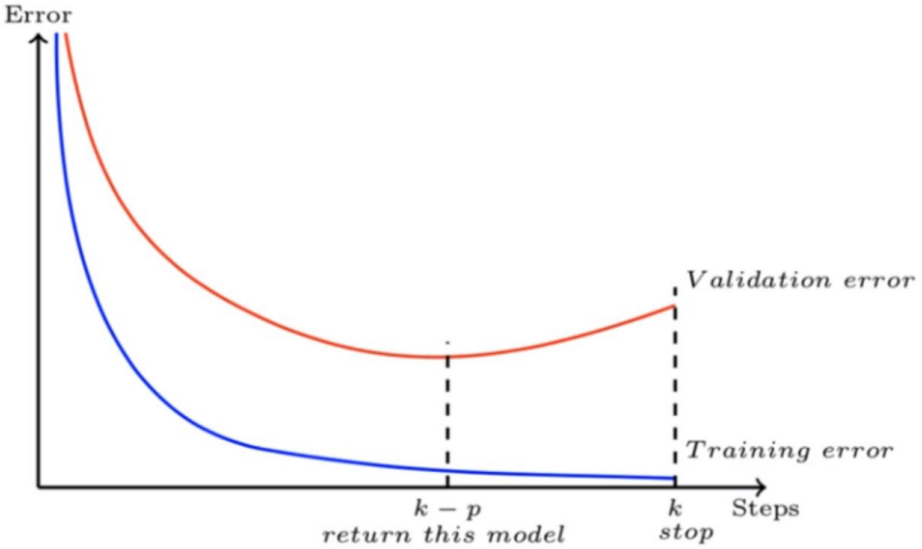
- 其中一個導致**Overfitting**的原因：沒有足夠的訓練資料
- 使用**Data Augmentation**可以解決這個問題
- 圖像資料特別適合使用**Data Augmentation**



Data Augmentation in Keras and Pytorch

- Keras和Pytorch都提供許多基本的Data Augmentation方法
  - 翻轉 (水平/垂直)
  - 亮度和對比
  - 旋轉
  - 縮放
  - 裁切
  - 扭曲
- 一般只會對訓練資料進行資料擴增
- Data Augmentation通常發生在Data Loader搬運資料時 (相當於每次讓模型看不同版本的圖片)

Early Stopping



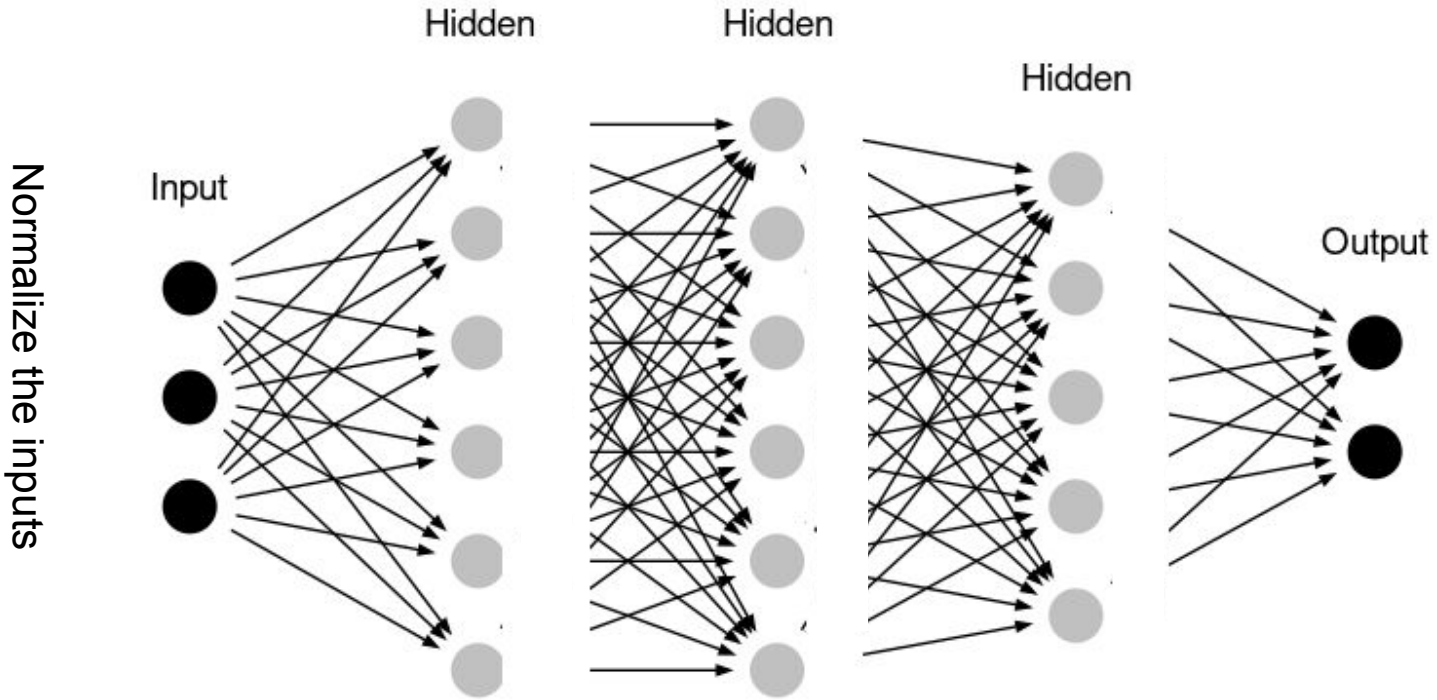
- 我們可以設定一個threshold，告訴模型如果在接下來的***P*** 個**Epochs**都沒有更好的結果，就停止訓練
- 在每個**Epoch**，我們都會儲存模型的權重

Early Stopping in Keras and Pytorch

- In Keras
  - 使用Callback
- In Pytorch
  - 需要手動實作

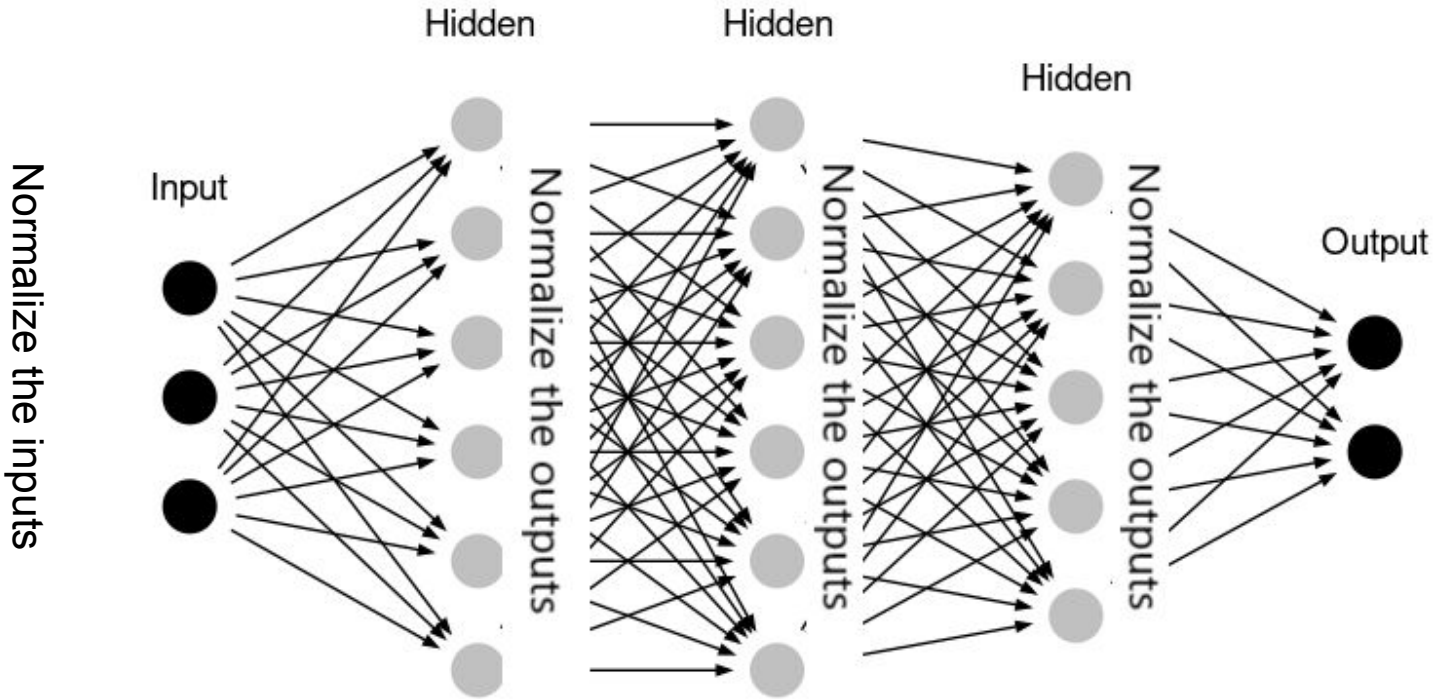
Source: https://csmoon-ml.com/index.php/2019/04/05/batch-normalization/

Batch Normalization



Source: https://csmoon-ml.com/index.php/2019/04/05/batch-normalization/

Batch Normalization



Batch Normalization怎麼運作

- 個別對每組Mini-Batch計算**mean**和**STD**以及進行標準化
- 針對CNN，每一層會有d個**means**和**STD**s (取決於有幾個filters)
- 通常會設在**Conv Layer**和**Activation Function Layer**之間
- 假如有使用**Dropout**，層的順序為：
  - Conv -> BacthNorm -> ReLU -> Dropout

Regularization小建議

- 不要一開始就使**用Regularization** (建立**Baseline Model**)
- 有時**Regularization**會對模型性能產生不利影響(例如，如果我們使用一些錯誤的參數設定)
- **Dropout**和**Batch Norm**會增加訓練時間
- **Dropout**: 不要在**Softmax Layer**的前一層使用
- 很簡單的模型不太需要使用**Regularization**
- 更多**Epochs**
- 如果**L2 Penalty**太高，模型可能會**Underfitting**

Implementation

Download notebook at:
https://github.com/albert831229/nchu-computer-vision/tree/main/113/day