**Support Vector Machines**

* O SVM é específico para classificação binário, encontra a decision boundary entre dois grupos, através dos Support vectors que mais separam os Support points. Enquanto optimiza a maior decision boundary, ou a seua margem, acaba por encontrar dois grupos e a semelhança entre os elementos de cada grupo. A operação matemática q permite achar esta divisão binaria é o dot product, no fundo há uma medida de semelhança a ser calculada entre dois grupos, em q se quer minimizar a semelhança dos Support points (esses deverão ser o mais diferentes)

• **What is Margin, and what it represents for the SVM algorithm**

* A distância entre o hyperplane e o ponto de dados mais próximo (pontos de suporte).
* A linha divide as duas classes de forma a que fique à mesma distância entre as duas classes através dos seus pontos de suporte.
* É a distância entre os pontos de suporte de cada classe.
* Quanto maior for a MARGIN melhor, para poder acomodar melhor novos dados.
* For SVM the optimal hyperplane is the one where the margin is maximized.
* The Margin is the distance of the vectors from the hyperplane.

• **How a Kernel works in solving non-seperable scenarios**

* Função transformadora que ajuda na separação.
* Recorre a um kernel trick, que através de algumas transformações cria novas “features” para encontrar uma fronteira não linear de decisão. Encontra uma forma de mapear em 3D para fazer a separação das classificações.

• **Common types of Kernels**

* Linear – Igual a non kernel
* Polynomial – função (Ver word Hugo)
* Gaussian -

• **Understand the way the SVM algorithm works and the steps it takes**

* SVM é um modelo de “supervised learning” com dados “labeled”. No processo de treino, o algoritmo analisa os dados de entrada e reconhece padrões espaço multidimensional denominado hiperplano. Todos os exemplos de entrada são representados como pontos neste espaço e são mapeados para categorias de saída de tal forma que as categorias ficam mais distintas.
* Como funciona a SVM? Uma máquina vectorial de suporte toma estes pontos de dados e produz o hiperplano (que em duas dimensões é simplesmente uma linha) que melhor separa as etiquetas. Esta linha é o limite de decisão: qualquer coisa que caia para um lado da mesma classificamos como azul, e qualquer coisa que caia para o outro como vermelho. (ajustar)
* A partir dos pontos do dataset produz um hiperplano (em 2d é simplesmente uma linha) que melhor separa as *label* (ou seja, melhora a classificação)
* SVM algorithm finds the optimal solution.

• **Advantages and disadvantages of SVMs**

* **Advantages**
  + Lida com facilidade com não linearidade
  + Utiliza programação quadrática (processo de resolução de funções quadráticas), com métodos específicos
  + Independentemente de quem corre o algoritmo, o resultado é sempre o mesmo
  + Lida facilmente com dados não balanceados através do reajuste dos pesos dos pontos
* **Desvantagens**
  + Os métodos são por vezes lentos
  + Não escala com facilidade para grandes datasets
  + Na prática a performance é pouco fiável
  + Modelos SVM são não interpretáveis.
  + A Escolha do kernel a utilizar pode não ser simples

• **What types of problems it can be applied in the ML space**

* Os svm são usados para aplicações como reconhecimento de texto, detecção de intrusão, detecção de rosto, classificação de email, calssificação de ADN e em páginas web. São vários motivos pelos quais usamos SVM em ML porque pode lidar com classificação e regressão em dados lineares e não lineares.

**Recommendation Systems**

• **Understand in which machine learning topic the Recommendation systems fits**

* Unsupervised Learning

• **Find the similarities and differences between the 2 algorithms given in theoretical classes**

* Ambos os algoritmos funcionam com base em semelhanças.

No caso do CBF, o sistema recomenda algo com base nas escolhas do user, procurando semelhanças entre os itens escolhidos por ele e todos os possíveis de recomendação.

No cado do CF, o sistema recomenda algo com base na semelhança entre 2 users

- Content based filtering (cbf): recomendação de algo semelhante com base no que o user já escolheu

- Collaborative filtering (cf)

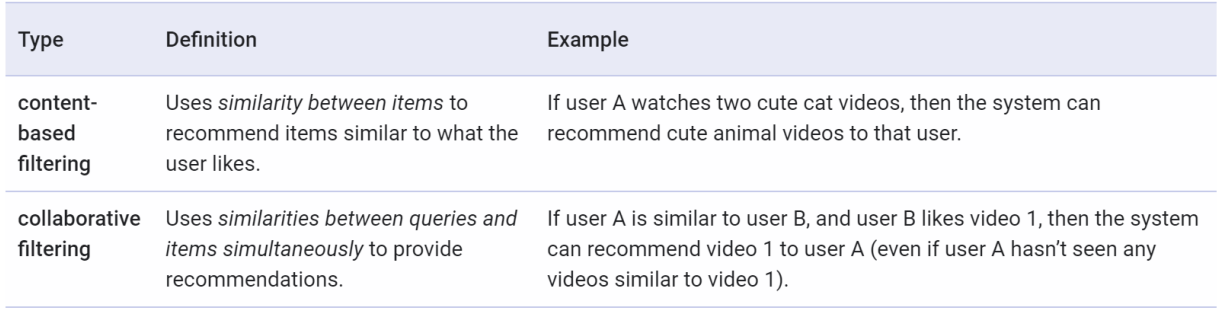
Statistical Method:

Item-item – User A viu o filme X, filme Y é semelhante ao X, então recomenda o filme Y ao user A

User-user – User A é semelhante (ratings/likes) ao user B, o user B viu o filme X, então recomenda o filme X ao user A

Model Method:

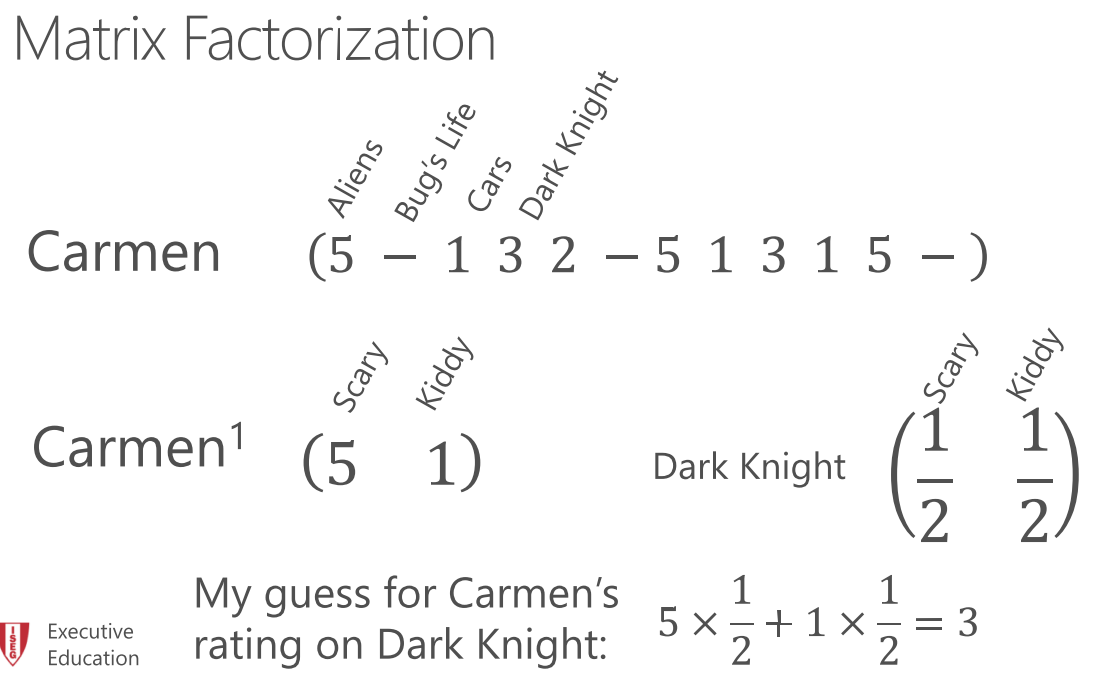
Singulas Value Decomposition (SVD)



• **Understand the input and output of each algorithm**

* **Content-filtering**: *Feature Matrix*
* **Collaborative-filtering**: Usa *Matrix Factorization*
* *Matriz Factoriztion* é uma forma de gerar características latentes ao multiplicar dois tipos diferentes de entidades. *Collaborative filrering* é a aplicação da *matrix factorization* para identificar a relação entre item e users
* **Collaborative** **Filtering**: is to discover the similarities on the user’s past behavior and make predictions to the user based on a similar preference with other users. This model is then used to predict items (or ratings for items) that the user may have an interest in.
* **Content-Based** **Filtering**: is used to produce items recommendation based on items’ characteristics.

• **Be able to perform a simple matrix factorization exercise**



• **Understand when to use one or other according to the problem given**

Exemplos:

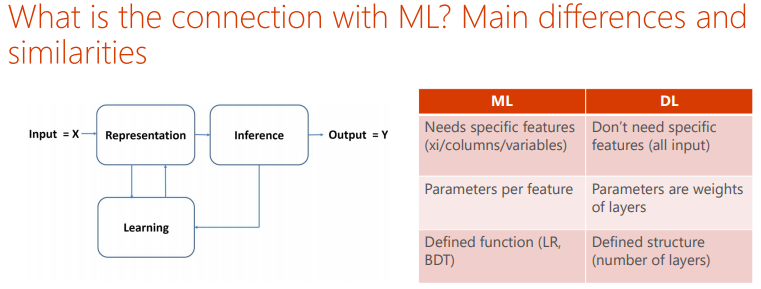
- Temos um conjunto de users que classificou um conjunto de filmes. Queremos recomendar um filme a um dos users que não classificou todos (no caso de ainda não ter visto). Através de um Collaborative Filtering (user-to-user) o sistema procura semelhanças entre os users (ratings) para recomendar um filme ao user em questão. User A é semelhante (ratings/likes) ao user B, o user B viu o filme X, então recomenda o filme X ao user A.

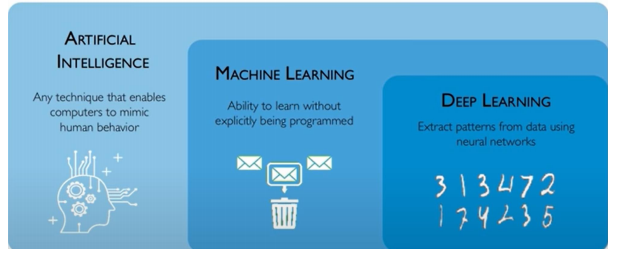
- Temos um user que classificou uma série de filmes. Queremos recomendar um filme que o user ainda não tenha visto. Através de um Content Based Filtering o sistema analisa o conteúdo dos filmes (features) e procura um novo filme que seja semelhante aos filmes que o user classificou melhor.

**Deep Learning Introduction**

• **Understand in which machine learning topic Deep Learning Techniques given fit**

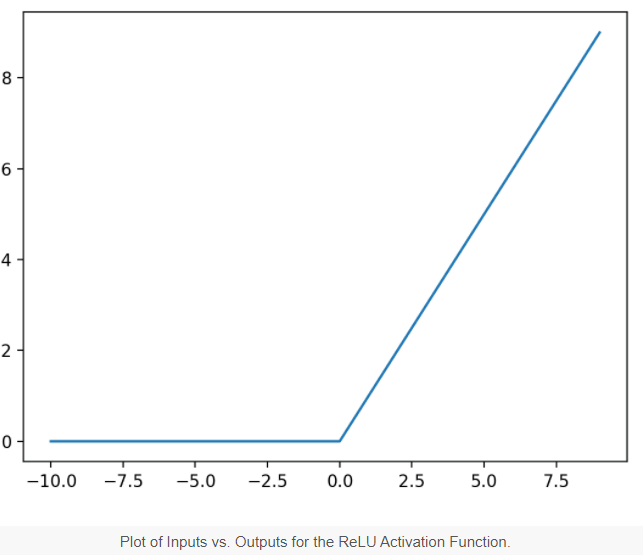
* **Deep Learning** é um tipo de Machine Learning (também visto como uma evolução de ML). ML trata-se da capacidade de os computadores pensarem e agirem com menos intervenção humana. No caso de DP, este segue o funcionamento do cérebro humano para processar datasets e tomar decisões eficientes. (*Multi-layered neutral networks*)
* DP é um subset de ML baseado em *Artificial Neural Networks*.
* Extrai padrões de dados através de redes neuronais
* Trata dados não estruturados, problemas com muitas dimensões porque aprende ‘features’ em vez de as contruir, e generaliza modelos independentemente do tópico.
* Grandes datasets (***Dimensionality***), ***variation***(muitas *features*) e **problemas semânticos** (NLP).



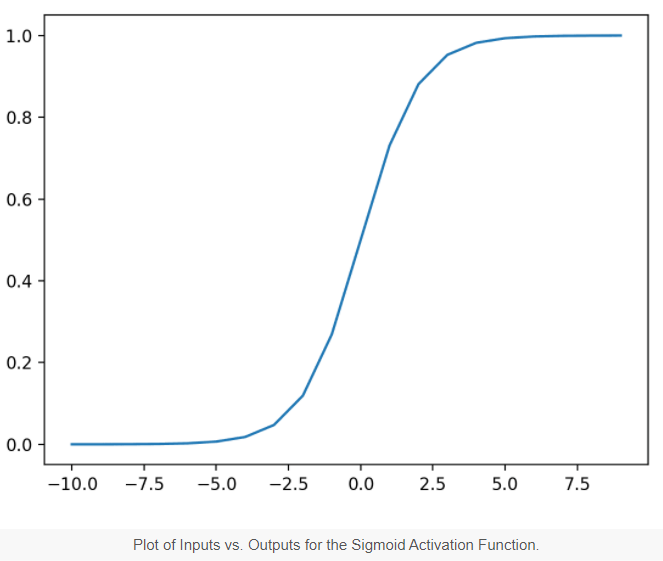


• **Understand how Activation functions work**

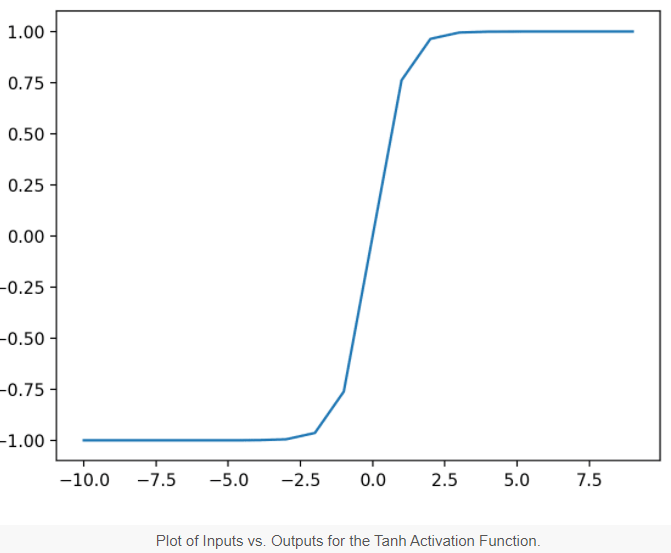
* ***Activation function*** controla a performance com que o modelo aprende com o training dataset, e define o tipo de previsão que o modelo consegue fazer.
* Define como os *weights* do input são transformados em outputs dos vários nós da *layer of the network*.
* Diferentes funções podem ser utilizadas em diferentes partes do modelo.
* É usada dentro ou após o processamento interno de cada nó da rede.
* Uma rede pode ter 3 tipos de camadas:
  + Input layers – que recebem o input
  + Hidden layers – que recebem input de outra camada e passam o output para outra camada (por norma com a mesma *activation function*
  + Output layers – realizam a previsão (*activion function* diferente das hidden que depende do tipo de previsão que se quer fazer)
* ***Activation functions***são tipicamente diferenciáveis. Significa pode ser calculado a derivada de primeira-ordem para um determinado *input*. Isto é necessário tendo em conta que que as *Neural Networks* são treinadas através da utilização de *bakcpropagation of error* que necessita da derivada de erro da previsão para actualizar os *weights* do modelo.

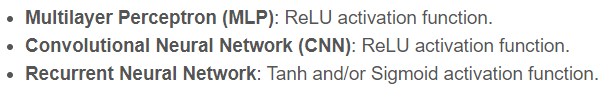


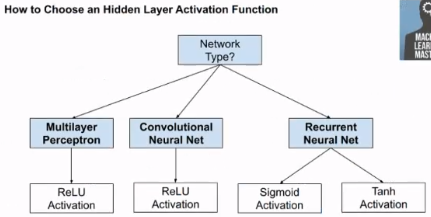
* **3 *Activation functions***nas *hidden layers:*
  + *ReLU* – simples e eficaz de implementar. Menos susceptível a “gradientes de fuga” que impedem o treino adequado do modelo. (se o valor do *input* for negativo, então é devolvido 0, caso contrário é devolvido o valor);

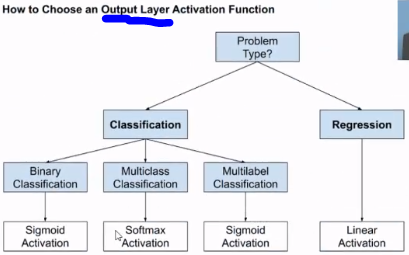


* + *Sigmoid (logistic function)* – mesma usada em Logistic regression classification algorithm. A função recebe qualquer valor real como *input* e dá *outputs* entre 0 e 1. Quando maior o *input* (mais positivo) mais perto o *output* será de 1. Quanto menor for o *input* (mais negativo) mais perto o *output* será de 0.
* *Tanh* – Semelhante à Sigmoid. A função recebe qualquer valor real de *input* e dá *outputs* de -1 a 1. Quanto maior for o *input* (mais positivo) mais próximo o *output* será de 1- Quanto menor for o *input* (mais negativo) mais próximo o *output* será de -1.









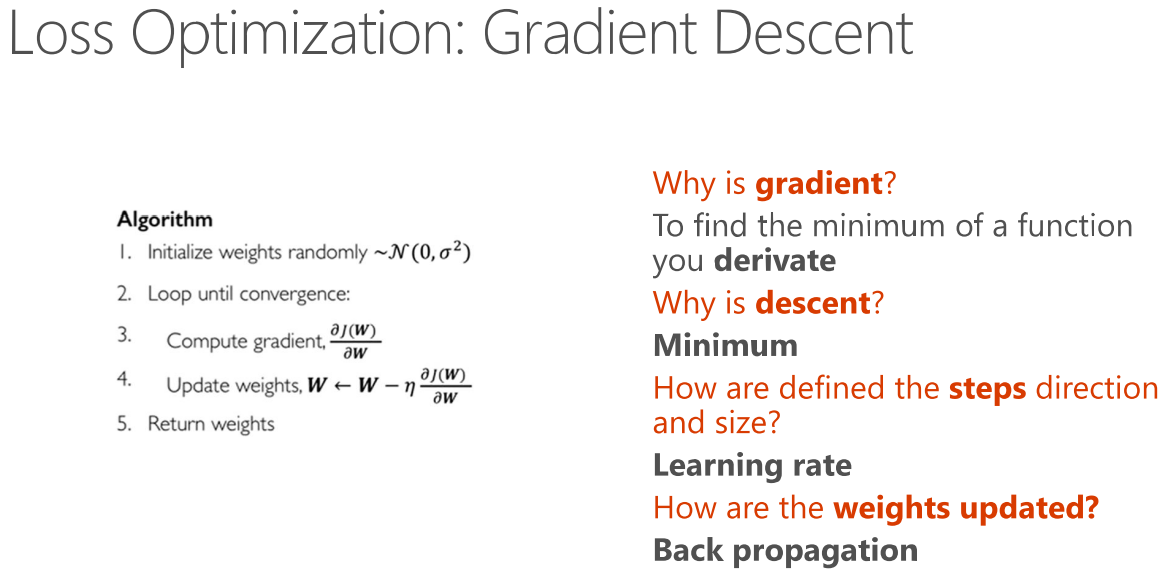
* *Activation function* nas *output layers*
  + Sigmoid (binaryclass zero ou um, multilabel ), softmax (multiclass), linear activation (regression)

***Multiclass classification*** *means a classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multiclass classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time.*

***Multilabel classification*** *assigns to each sample a set of target labels. This can be thought of as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A text might be about any of religion, politics, finance or education at the same time or none of these.*

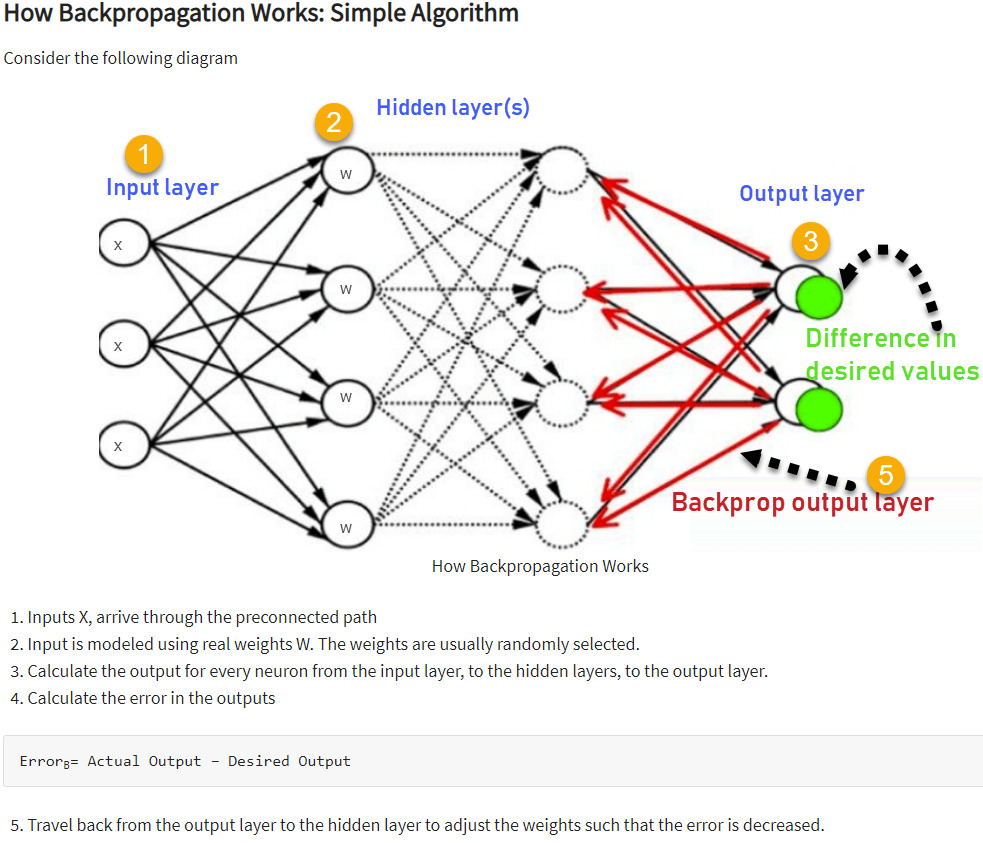
• **Understand Gradient Descent and how it works** (tem de se usar uma loss function, é o que estamos a medir)

* É um método numérico (algoritmo) utilizado para encontrar os valores dos parâmetros (coeficientes) de uma função que minimiza uma função de custo. O objectivo é continuamente tentar diferentes valores para os coeficientes, avaliar o seu custo e seleccionar novos coeficientes que tenham um custo ligeiramente melhor (custo mais baixo). A repetição deste processo vezes suficientes pode levar a encontrar os valores do coeficiente que resultam no custo mínimo.
* Procedimento:
  + Começa com valores iniciais para os coeficientes da função.
  + O custo dos coeficientes é avaliado ao liga-los à função e calculando o seu custo.
  + Calcula-se a derivada do custo (referente à inclinação da função num determinado ponto). Precisamos de saber qual a inclinação para conseguirmos saber qual a direcção (sinal) para onde devemos mover os valores do coeficiente de modo a obter um custo mais baixo na iteração seguinte.
  + Após sabermos, através da derivada, qual a direcção podemos actualizar os valores dos coeficientes. Deve ser especificado um parâmetro de *learning rate* (*alpha*) que controla quanto os coeficientes podem alterar em cada actualização.
  + O processo repete-se até que o custo dos coeficientes seja de 0 ou suficientemente próximo de zero para ser considerado bom.



• **Understand Back propagation and how it works** (tem de se usar uma loss function, é o que estamos a medir)

* Algoritmo que treina os pesos de uma rede neural
* Exige que propaguemos a informação para trás através da rede, depois para a frente, para trás, e assim sucessivamente.
* É o método para ajustar os *weights* da *neural net* com base na taxa de erro obtida na iteração anterior. O ajuste correcto dos *weights* permite reduzir a taxa de erro e torna o modela mais fiável.

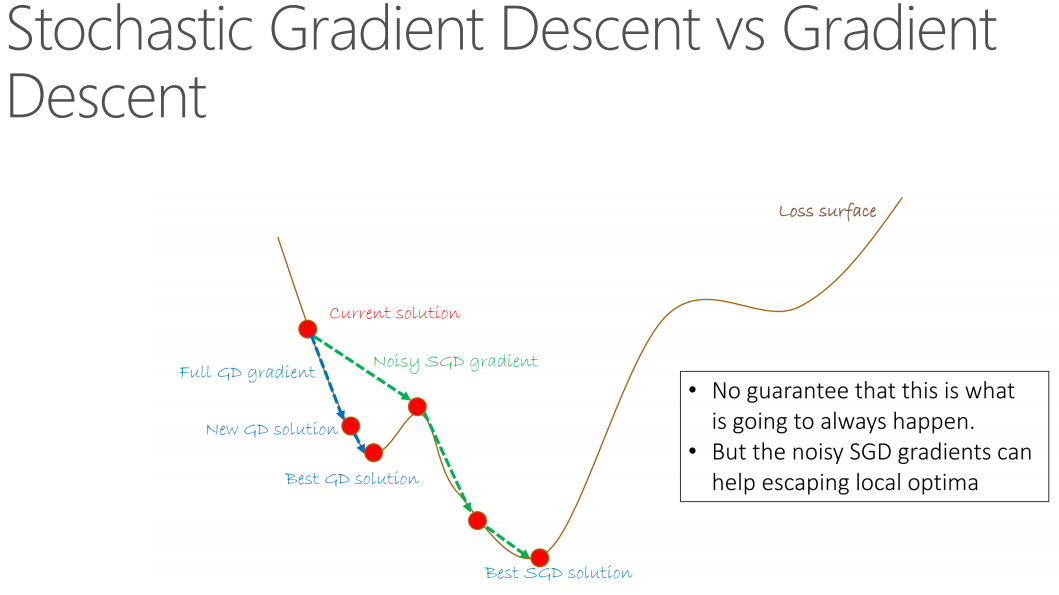


• Understand how weights are propagated and how the training of the models is done

**Optimization techniques in Deep Learning**

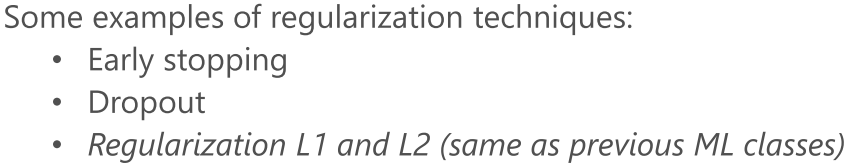
• **Differences in the gradient descent alternatives given**

* ***Gradient Descent*** *(Bom para problemas convexos, encontrando a local mínima, e não a global)*
  + Todas as *training samples* são o representante absoluto da distribuição dos *inputs*
  + Adequado para problemas de optimização tradicionais: “encontrar a melhor rota”
  + Mas em Ml não podemos fazer esta suposição, tendo em conta que os dados para teste são sempre diferentes
  + Em datasets reais a *GD* não é realista
* ***Stochastic Gradient Descent***(*SGD*) – técnica melhorada da GD, permite encontrar o mínimo global.
  + *One set of training sample to do the update for a parameter in a particular iteration*
  + O modelo não se ajusta demasiado às *training samples*
  + Pode fazer cálculo em paralelo
  + Pode ajudar as escapar aos mínimos locais
  + Rápido de GPU
  + As soluções mais generalistas
* ***Mini-batch Gradiente Descent*** – divide o dataset em pequenos *batches* (lotes) que são utilizados para calcular o erro (através da *Gradient descent*) e actualizar os coeficientes do modelo.

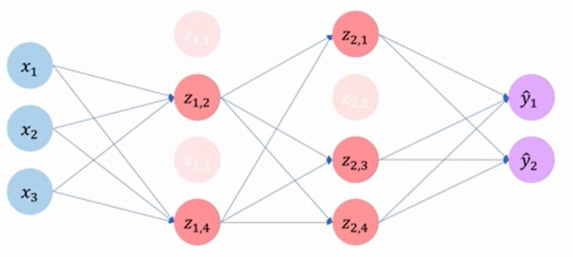


• **Identify the need for regularization and the approaches available**

* Regularização é um método que se aplica quando o treino está ***overfitting***.



* **Drop out** – é uma técnica de definir a *activation* a zero.



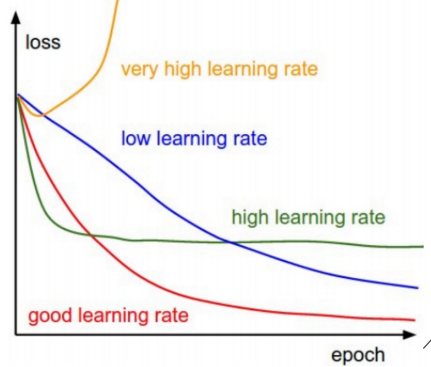
* + Reduz as co-dependências complexas entre *neurons*
  + Cada *neuron* torna-se mais robusto (generaliza melhor)
  + Diminui o *overfit* (consequência)
* **Early stoping** – é uma técnica que consiste em parar o *training* quando a perda no *training* e no *test* é divergente

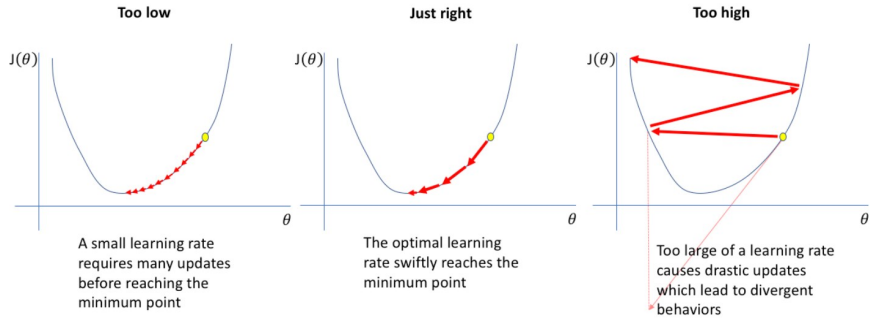


* + Monitorizar a performance num set de validação separado
  + *Training* da rede diminuirá o *training error*, bem como o erro de validação
  + Para quando o erro de validação começa a aumentar (overfit)

• **Understand the purpose of the learning rate** (convergir mais rápido, e é feito por *weight*)

* A *learning rate* é um *hyperparameter* utilizado para controlar a rapidez com que o modelo é adaptado ao problema em questão. *Learning rates* mais pequenas requerem mais *training epochs* (tendo em conta as alterações mais pequenas dos *weights* em cada actualização), *learning rates* mais maiores resultam em alterações mais rápidas e requerem menos *training epochs*.
* Controla a *rate* à qual um algoritmo actualiza o parâmetro ou aprende os valores dos parâmetros.
* Determina o tamanho do passo de cada iteração enquanto se move para o mínimo da *loss function*. Representa a velocidade a que um modelo “aprende”.
* Uma *learning rate* demasiado alta pode fazer com que o passo a dar ultrapasse o minínimo, e uma *learning rate* demasiado baixa vai demorar mais tempo a convergir ou parar a aprendizagem num mínimo “não optimo”.





**Image Processing /CNN**

1. **Supervised Learning**

• **Understand intuitively the convolution technique (mainly with matrix numbers)**

* A *convolution technique* recorre a uma convulsão (operação matemática entre duas matrizes para produzir uma terceira) para extrair detalhes importantes das imagens. Esta operação usa um filtro (kernel) que percorre toda a imagem mediante o *stride* definido, resultando num conjunto mais de imagens mais pequenas já que a cada momento que o filtro é aplicado resulta numa imagem do tamanho desse filtro com o resultado da convulsão aplicada.

• **Understand the convolution characteristics and why are CNN used in images**

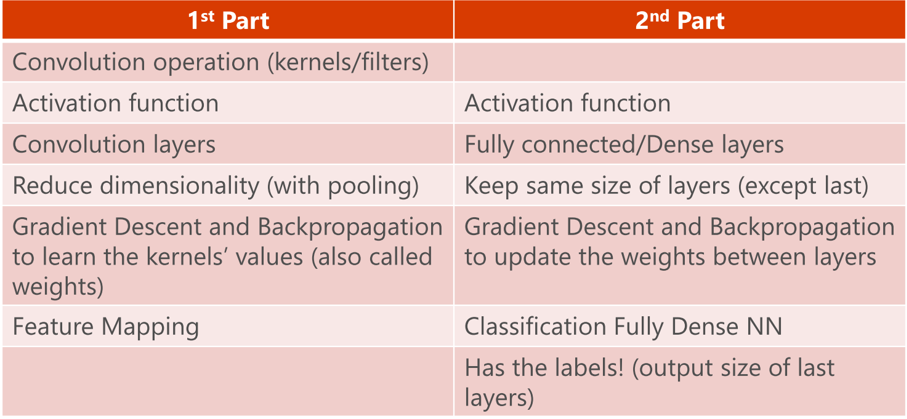
* CNN consegue aprender *informative features* das imagens, eliminando a necessidade do tradicional processamento manual de imagens
* Redes flexíveis
* Exploram o facto de um padrão interessante poder ocorrer em qualquer região da imagem.
* O modelo aprende *features* úteis a partir de dados em bruto
* Usa 2D *convolutional layers* (ideal para processar dados 2D, imagens)
* Elimina a necessidade de extracção manual de *features*
* Extrai as *features* directamente das imagens, e as *key features* são aprendidas durante o treino da CNN numa colecção de imagens
* Adequado para classificação de objectos/imagens
* CNN são usadas para aprender *feature maps* complexos
* Invariável à tradução e distorção de *features*
* Reduzir a dimensionalidade dos *inputs tensors*
* Partilham os *weights* e são relativamente fáceis de treinar
* Parameter Sharing
  + O *kernel* (com os *weights*) usa os pesos pela imagem toda
  + Os *weights* são treinados ao usar métodos de *backpropagation* e *gradient* *descent*
  + CNN usa *parameter sharing*
* **Epoch** indica o nº de ciclos completos que a rede neuronal executa utilizando todo o dataset de treino (ex. 3 epochs a rede treinou 3 vezes com todo o conjunto de imagens de treino que tem.
* **Batch size** é o nº de amostras de treino que vão ser utilizadas durante a iteração de treino. Na análise do *batch* os valores previstos serão comparados com os reais e o erro será calculado (a passagem pela rede e o backpropagation é feito por batch)

• **Being able to calculate filtering dimensions**

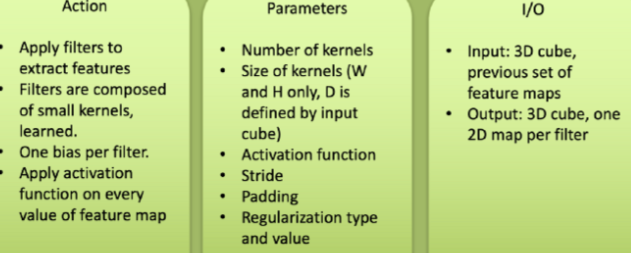
* FICHEIRO EXCEL RUI

• **Understand characteristics and purpose of 1st part and 2nd part of the CNNs**

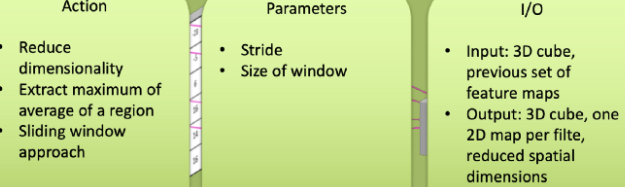
* **1st part – Features mapping**
  + Extrai as *features* que se destacam das imagens num sentido do mais concreto para o mais abstrato
* **2nd part – Classificação**
  + Permite determinar a probabilidade de uma dada imagem pertencer a uma classe. Nesta fase temos uma rede fully connected.



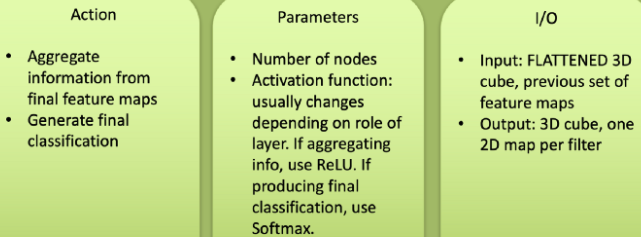
* Kernels/filters – detectam padrões
  + Detecção de ângulos, figuras geométricas, olhos, *edge detection*, *sharpen*, etc
  + Reduz a quantidade de *weights* que a CNN tem de aprender, e quando a localização destas *features* altera não acaba com a CNN
  + Filtros de 3x3 por exemplo, vai correr a imagem em todos os 3x3
  + Quanto mais para a frente nas layers mais complexos são os filtros, e mais tipos de padrões conseguem detectar
* Activation function
* Convolution layers

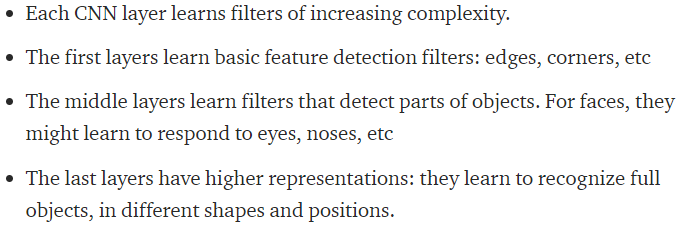


* Gradient desent and backpropagation lo learn weights values
* Feature maping – é o resultado da aplicação de um filtro/kernel numa image de *input*.
  + É o *output* de cada *layer*.
  + A visualização do *feature map* ajuda a compreender as *features* que a CNN detectou
* Pooling layers (reduce dimensionality)

c

* Fully connected layers





**Text Processing / LSTM**

• **Understand the main characteristics of Recurrent Neural Networks**

* Supervised learning
* Tem memória interna
* *Sequencial data*
* Usa a informação anterior para produzir o *output* seguinte
* Desempenha a mesma função para cada *input*, enquanto o *output* depende do cálculo anterior. Após produzir o *output*, este é copiado e enviado de volta para a rede.
* Para a tomada de decisão é considerado o *input* e o *output* que aprendeu anteriormente
* Temos loops, ou seja, o *output* de uma unidade pode voltar a uma unidade anterior
* Pode utilizar a sua memória para processar sequências de *inputs*

• **Identify, given a problem, it’s application**

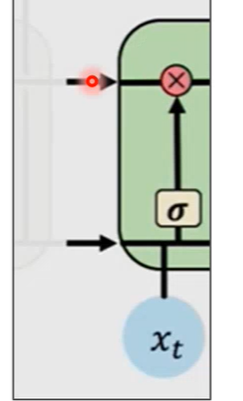
* Pode ser utilizado para *regression* (prever o valor da stock com base numa time serie)
* Pode ser utilizado para *classification* (prever a próxima palavra num texto, ou a próxima nota numa música)
* Necessita sempre de uma *label*, está dentro de *Supervised Machine Learning*
* Problema de sequências
* Vídeos
* Time series data
* Biological measurements
* Climate measurements
* Market analysis
* Speech/music/texto
* User behavior in websites

• **Understand the differences (and similarities) between RNN and LSTM**

* Ambos utilizadas para sequence-sequence problemas
* LTSM é um tipo de RN, onde as várias unidades se conectam numa forma específica para evitar problemas das RNN regulares como o *vanishing* e o *exploding gradiente*)
* *Classify, process and predict*
* Treina o modelo usando back-propagation

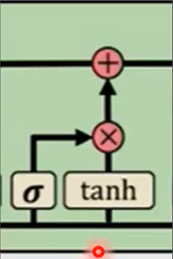
• **Know each component (input/output/purpose) of LSTM**

* LSTM



Forget

* + Forget
    - O input é somado ao output da anterior, depois passa pela sigmoid. O resultado é multiplicado pelo output da anterior. No caso de resultado da segmoid ser 0 passamos a esquecer o output da anterior.

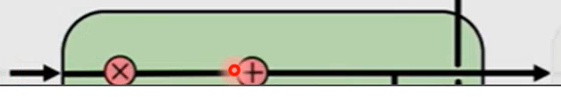


Store

* + Store
    - Pega no input e faz uma sigmoid para decidir se guarda o valor. Passa também por uma activation function (tanh) e multiplica com o resultado

da sigmoid. Posteriormente soma com o resultado do primeiro Forget.

* + Update



Update

* + - O que vem do Forget e do store, ou seja, mantém o que e importante
  + Output
    - Mantém apenas o que é importante, do input inicial(com uma sigmoid) e do update (com um tanh)



Gaudssian white noise

independet

**Time Series**

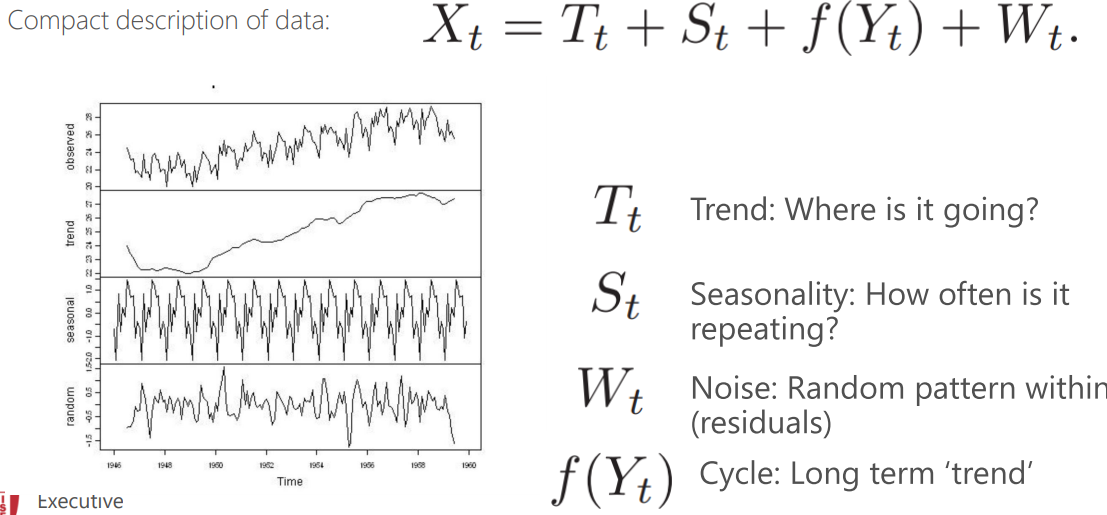
• Understanding of each time series’ characteristics

* *Stochastic* – eventos *random* (Vai chover hoje? Vai sair o nº2 no dado lançado?)
* *independent* – *white noise*
  + eventos independentes entre si (o oposto de *time series*)
* *Dependence*



Dependent

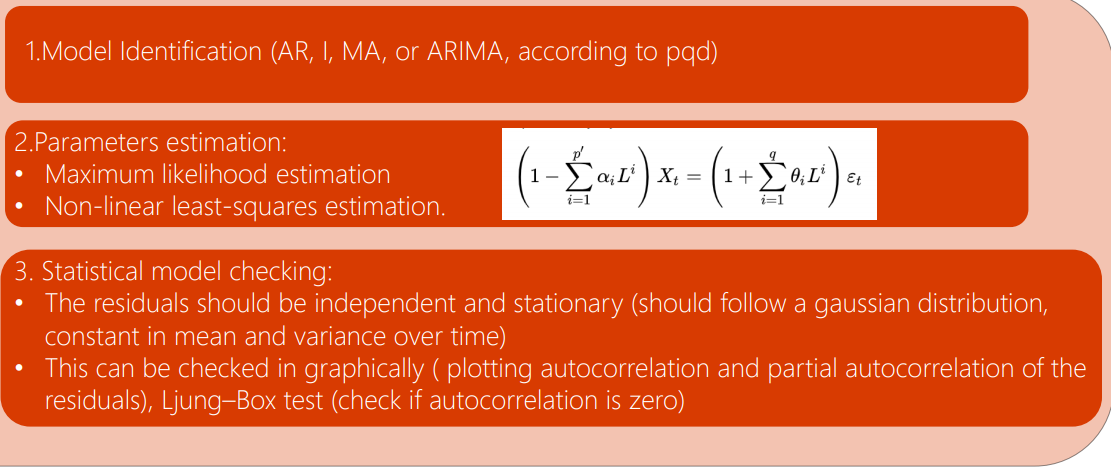
* + Em *time series* cada nova observação é *stochastically* dependente da anterior.
  + Dependência é boa porque permite fazer previsões
  + Dependência é má porque os pressupostos têm de ser corrigidos (teorema do limete central, deve ser corrigido para *dependente observations*
* Dados com alta frequência
* Tempo no eixo dos xx
* Dados dependentes
* Dados sequenciais
* Objectivos
  + Interpretação – ajustes por épocas
  + Forecasting – prever o desemprego
  + Controlo – impacto das políticas ficais no desemprego
  + Teste de hipóteses – Aquecimento global
  + Simulação – estimar probabilidades sob condições pré-concebidas



* + Componentes
    - Tendência – aumento ou diminuição a longo prazo (tendência ascendente ou descendente)
    - Sazonalidade – flutuação periódica dentro de um certo período de tempo. Padrão que tende a repetir-se num período sazonal seguinte
    - Ciclo – ao longo de um grande intervalo de tempo
    - Movimento irregular (componente/noise/residuals) – o movimento após se explicar a tendência, sazonalidade e ciclos; ruído aleatório ou erro

• Understand the components of ARIMA models

* *Supervised forecast* – previsão dos valores seguintes numa janela temporal
  + Valores de time series são as label
  + Tópico de regression (diz-se forecasting)
* Para aplica-lo estamos sob s suposição uma *time serie* estacionária
* **AR – autoregressão**
  + Modelo que utiliza a relação dependente entre uma observação e um certo nº de observações desfasadas
  + **p** – nº de observações de atraso (*lag*) incluídas no modelo (*lag order*)
* **I – integrado**
  + Utilização de diferentes observações em bruto (ex., subtrair uma observação à observação anterior temporalmente) para tornar a série estacionária
  + **d** – nº de vezes que as observações *raw* são diferentes, grau de divergência (*degree of differencing*)
* **MA – Média móvel**
  + Modelo que utiliza a dependência entre uma observação e um erro residual de um modelo médio aplicado a observações desfasadas
  + **q** – tamanho da “janela da média móvel”, *order of moving average*



ARIMA Model: Box-Jenkins method

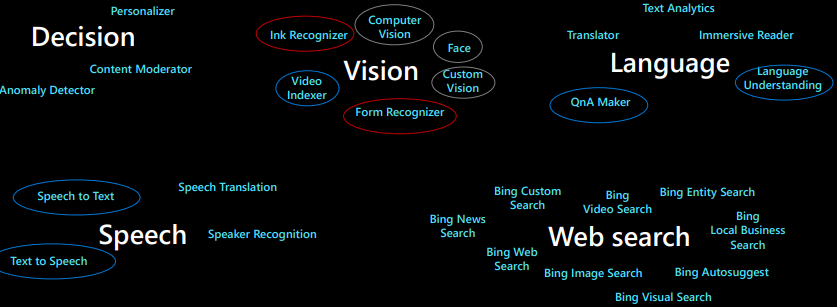
• Understand when we are within time series forecast or anomaly detection

* Exemplo: agora são 10h da manhã e quero saber se vou ter anomalias no meu ar condicionado durante o dia (forecast)
* Agora são 10h da manhã e o detection avisa me que está a acontecer uma anomia
* **Anomalia detection**
  + Series temporárias
  + Dataset normal com uma pequena parte de anomalia (até 10% do dataset)
  + UNsupervised ML

**Built-in AI**

• Know the types of built in models that were presented

• Understand when to use each type of service



• Understand the concept of Transfer Learning and how it enabled the possibility of

these services to be created and customized

* O conceito de *transfer learning* acontece quando pegamos num modelo de ML já treinado para uma acção/detecção/classificação específica e usamos o seu conhecimento adquirido para aplicar noutro problema diferente, mas “relacionado”. (ex. temos um modelo de classificação treinado para prever se uma imagem contém uma mochila, podemos usar o conhecimento que esse modelo adquiriu durante o treino para fazer com que outro modelo possa detectar outro objectos como uns óculos).
* A ideia é o utilizar o conhecimento adquirido de um modelo treinado com várias *labels* para uma nova tarefa que não tenho um dataset tão bom (tamanho). Em vez de começar um novo modelo do zero, aproveitamos o reconhecimento de padrões já existentes do anterior para ajudar a resolver a nova tarefa.
* É uma metodologia de design.
* No ex. dos óculos e mochila. Nas primeiras *layers* o modelo aprende a detectar objectos nas imagens, e só nas *layers* seguintes é que o modelo vai tentar detectar o que é mochila (objecto pretendido) em comparação com os restantes obejctos na imagem. Tendo isso em conta podemos usar estas primeiras *layers* para iniciar outro modelo em que as *layers* seguintes tentam separar os vários objectos na imagem e detectem os óculos.
* Vantagens
  + Poupança de tempo em treino de modelos
  + Melhor performance de *neural networks*
  + Menor necessidade de grandes datasets
    - Podemos construir um modelo com datasets curtos porque a parte que estamos a utilizar de outro modelo (*transfer learning* de outro modelo) já está pré-treinada. (ideal para processamento de linguagem, porque necessita de muito conhecimento específico para criar grandes quantidades de *labeled datasets*

• Understand the Differences between Object Detection and Image Classification

* *Object detection* é quando detectamos que existe um objecto específico na imagem;
* *Classification* é quando na apresentação de uma imagem ele classifica os objectos (por ex. cão ou gato)

• Understand the importance of Acoustic and Linguistic Models in the accuracy of the ASR Models

* ***Acoustic Model*** – é um classificador que rotula pequenos fragmentos de áudio em uma de várias unidades de som de uma dada língua
  + Requer uma colecção de dados de fala
    - Set de áudio files
    - Texto de transcrições de cada áudio file
  + Os dados de áudio devem representar o mesmo cenário onde queremos utilizar. Ou seja, se queremos reconhecer um discurso num ambiente com barulho, os dados de áudio devem conter pessoas a falar num ambiente do mesmo tipo.
* ***Language Model*** – Distribuição de probabilidades por sequências de palavras. Decide entre sequências de palavras que soam semelhante, com base na probabilidade da própria sequência da palavra.
  + Semelhante a acoustic model mas apenas com texto
  + Ficheiro de texto contém lista de afirmações para o sistema
  + Não precisam de estar completas ou gramaticamente correctas
  + Reflecte com precisão o input falado que se espera que o sistema encontre
  + Os exemplos devem reflectir o estilo e o conteúdo da tarefa que o utilizador irá realizar com a aplicação
  + Um exemplo (frase, enunciado ou pergunta) por linha
  + Se quisermos algumas frases com maior peso (importância), podemos adicioná-las várias vezes aos dados (bom entre 10-100)

Acoustice model e Language model são importantes para os modelos de ASR porque se complementam de maneira a conseguir capturar a totalidade das propriedades do áudio. Ou seja, o Acoustic model capta o áudio e o texto respectivo e posteriormente o language model ajuda na transformação correcta mediante o contexto em questão.

• Understand what are NLU models and the elements that it identifies from text inputs

and how they are trained