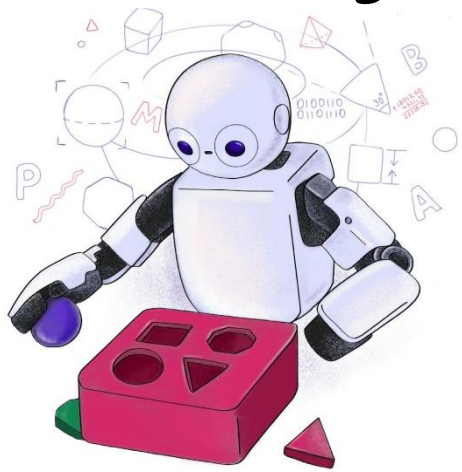
TP557 - Tópicos avançados em IoT e Machine Learning:

Prevenindo o sobreajuste



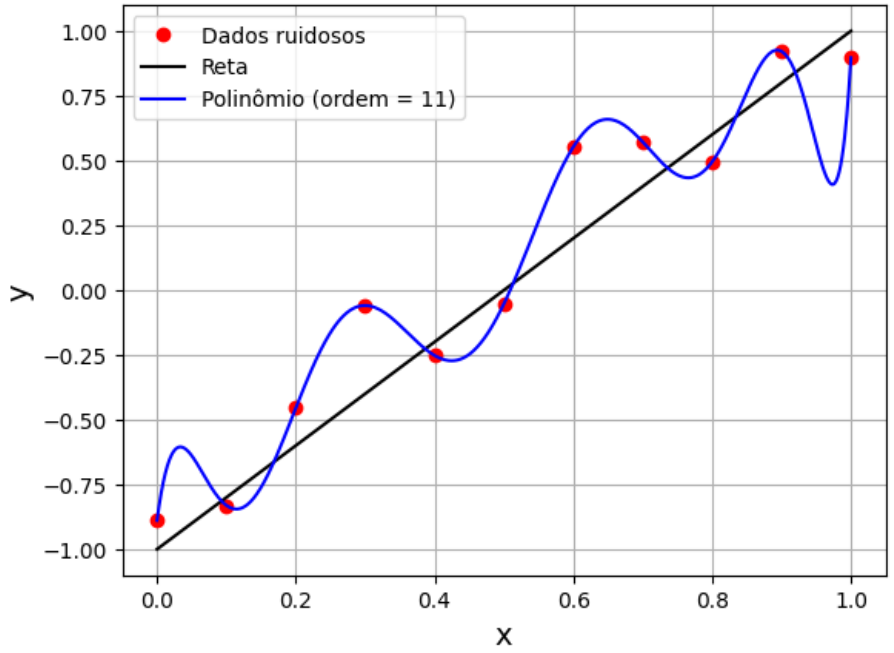
Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

O que vamos ver?

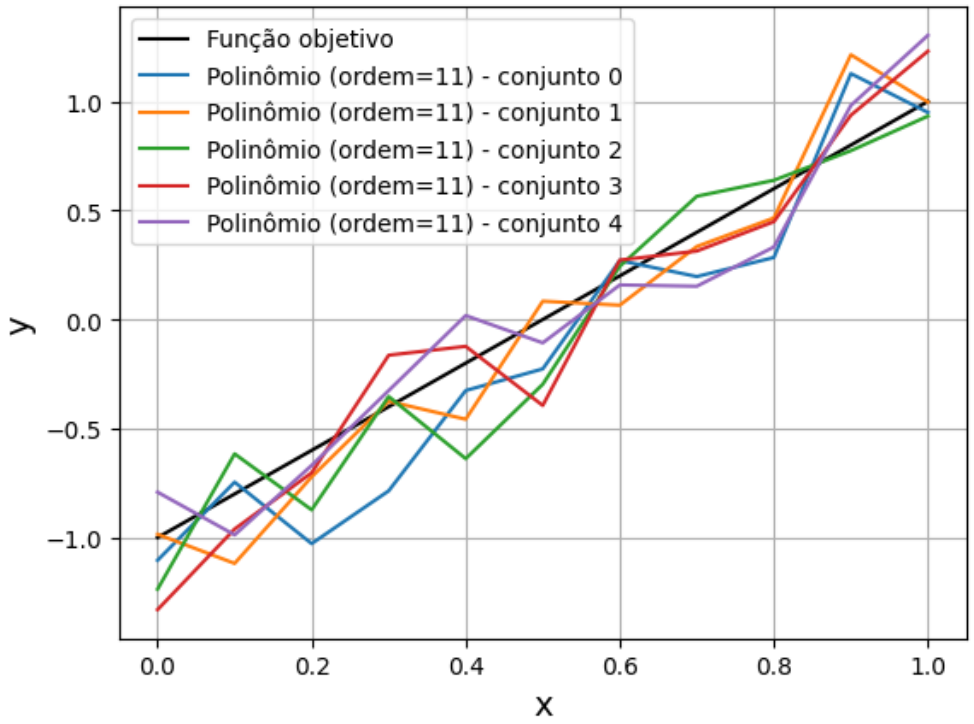
- Vimos anteriormente que o sobreajuste (*overfitting*) é um fenômeno indesejado que ocorre em modelos de aprendizado de máquina, no qual o modelo se ***ajusta excessivamente aos dados de treinamento***, em vez de generalizar bem para dados inéditos.
- Em outras palavras, o ***modelo "decora" os dados de treinamento capturando até mesmo o ruído presente nos dados*** em vez de aprender o padrão geral que pode ser aplicado a exemplos desconhecidos.
- Em geral, é ***mais comum um modelo*** se ***sobreajustar*** do que subajustar.
- Portanto, neste tópico, veremos algumas formas de se evitar o sobreajuste.

Sobreajuste



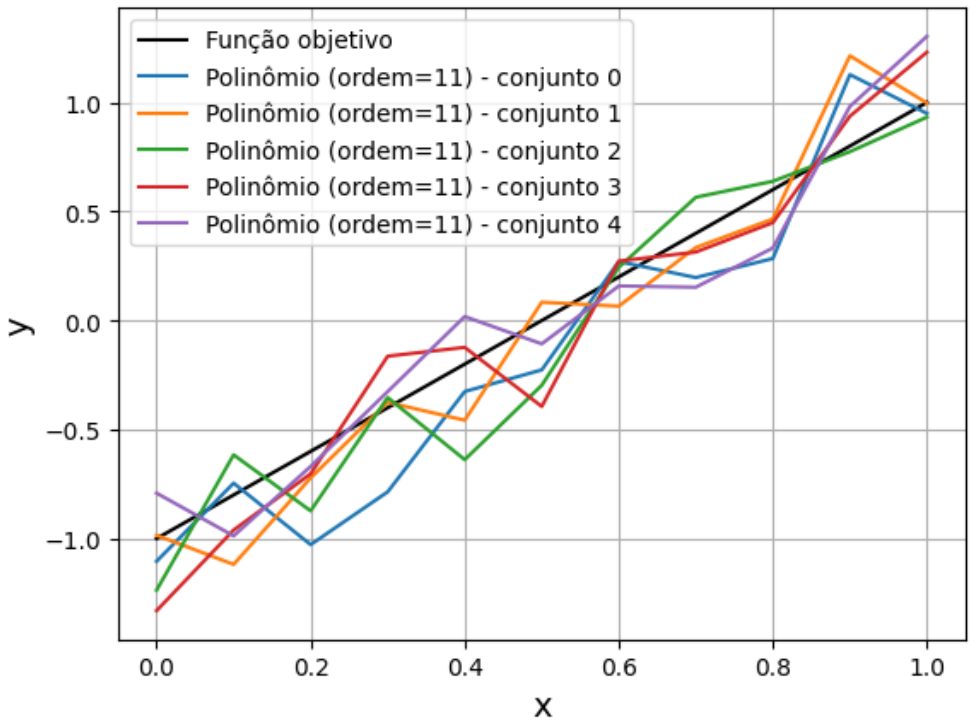
- A figura ao lado mostra 11 ***amostras ruidosas*** geradas a partir de uma reta, chamada de ***função objetivo***.
- Elas são aproximadas por uma ***reta*** e um ***polinômio de ordem 11***.
- Embora o ***polinômio se ajuste perfeitamente*** aos dados ruidosos, pode-se esperar que a ***reta generalize melhor***.
- Se as duas funções forem usadas para fazer predições além das 11 amostras, a ***reta*** deve obter ***melhores resultados no sentido da minimização do erro médio***.

Sensibilidade aos dados de treinamento



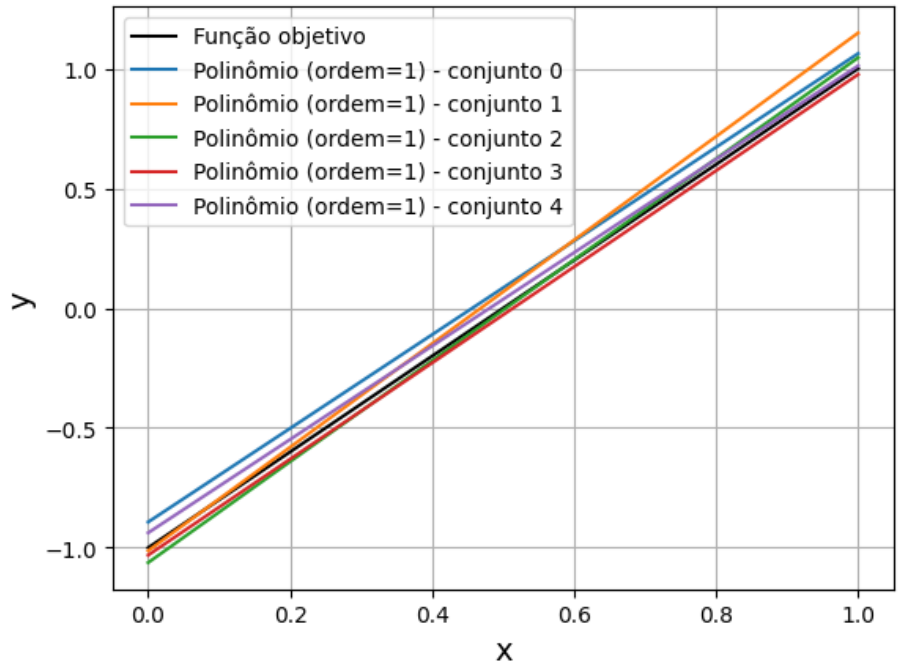
- Em essência, o ***sobreajuste*** faz com que um ***modelo extraia***, sem saber, parte da ***variação residual***, ***como se essa variação representasse o padrão geral por trás dos dados***.
  - ***Variação residual*** são ***padrões que não são significativos***, como por exemplo o ruído presente nas amostras.
- Um modelo ***muito flexível***, que se sobreajusta aos dados de treinamento, apresenta ***alta variância***.

Sensibilidade aos dados de treinamento



- A alta variância significa que o ***modelo*** é muito ***sensível às variações nos dados de treinamento***.
  - O ***modelo será diferente para cada conjunto de treinamento***.
  - Nesse caso, o modelo se distancia do padrão geral por trás dos dados.
- No exemplo ao lado ele irá se ***ajustar tão bem aos dados que vai aprender até o ruído presente nas amostras***.
  - Cada conjunto de treinamento ***corrompido com amostras ruidosas diferentes resultará em um modelo diferente***.

Complexidade (ou flexibilidade) ideal



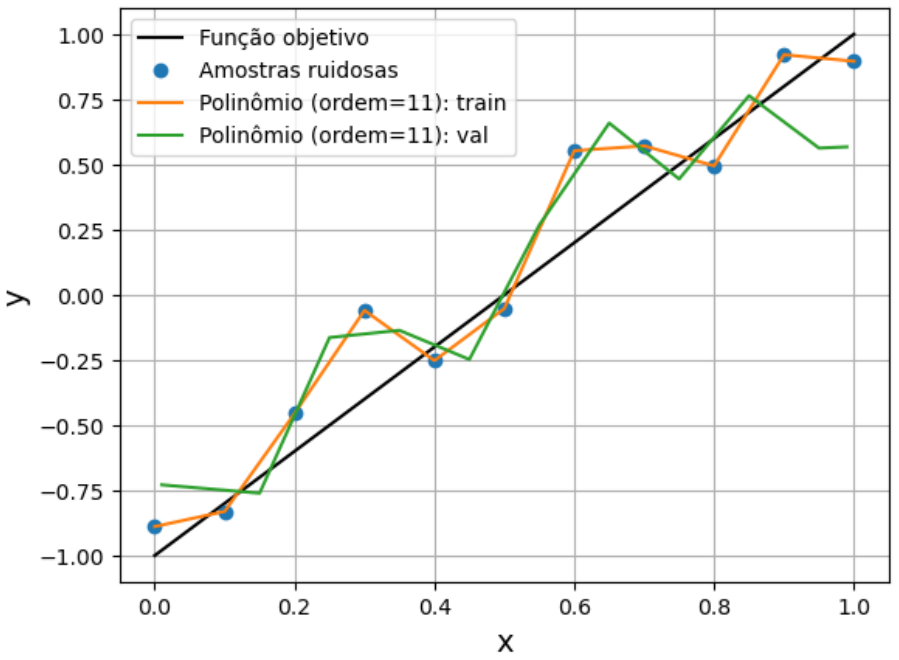
- O modelo mostrado na figura ao lado tem ordem apropriada (i.e., ordem 1).
- Percebam que ele ***não varia tanto*** como no exemplo anterior ***quando se varia o ruído adicionado às amostras de treinamento***.
- Um modelo que apresente ***flexibilidade ideal*** terá ***baixa variância***, ou seja, ele sempre ***tenderá a capturar o padrão geral por trás das amostras***, ***mesmo que ruidosas***.

Exemplo: Overfitting.ipynb

Causas do sobreajuste

- **Modelo complexo:** Um modelo ***muito complexo***, com muitos parâmetros (i.e., pesos), tem uma ***alta capacidade de representação*** (ou flexibilidade) e pode ***se ajustar demais aos dados***.
- **Dados insuficientes:** Quando os ***dados de treinamento são limitados em quantidade***, o ***modelo pode não ter informações suficientes para generalizar*** adequadamente.
- **Ruído nos dados:** A presença de ***ruído ou outliers nos dados de treinamento*** pode fazer com que o ***modelo tente ajustar-se a essas variações aleatórias***.
  - O ruído é chamado de erro irredutível, pois dificilmente nos livramos dele.

Como detectar o sobreajuste

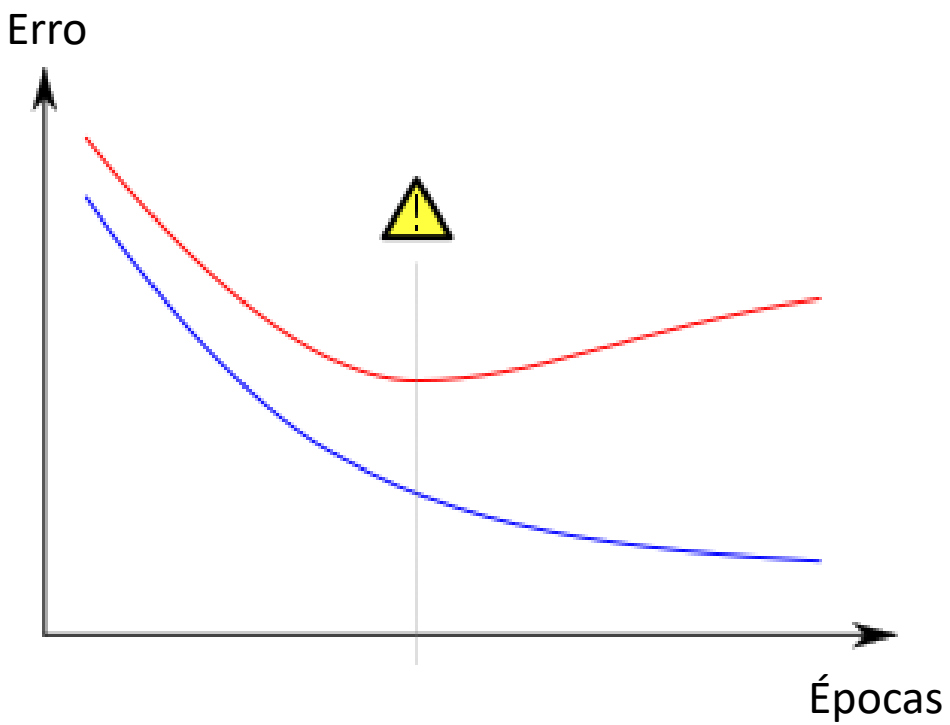


MSE train: 3.023264662910588e-24
MSE val: 0.18213360918420743

- **Conjunto de Validação:** ***Dividir o conjunto total*** de em um conjunto de treinamento e um conjunto de validação e ***avaliar o erro em ambos os conjuntos ao longo do treinamento***.
- Se o ***desempenho no conjunto de validação for significativamente pior*** do que no conjunto de treinamento, é um ***sinal de possível sobreajuste***.

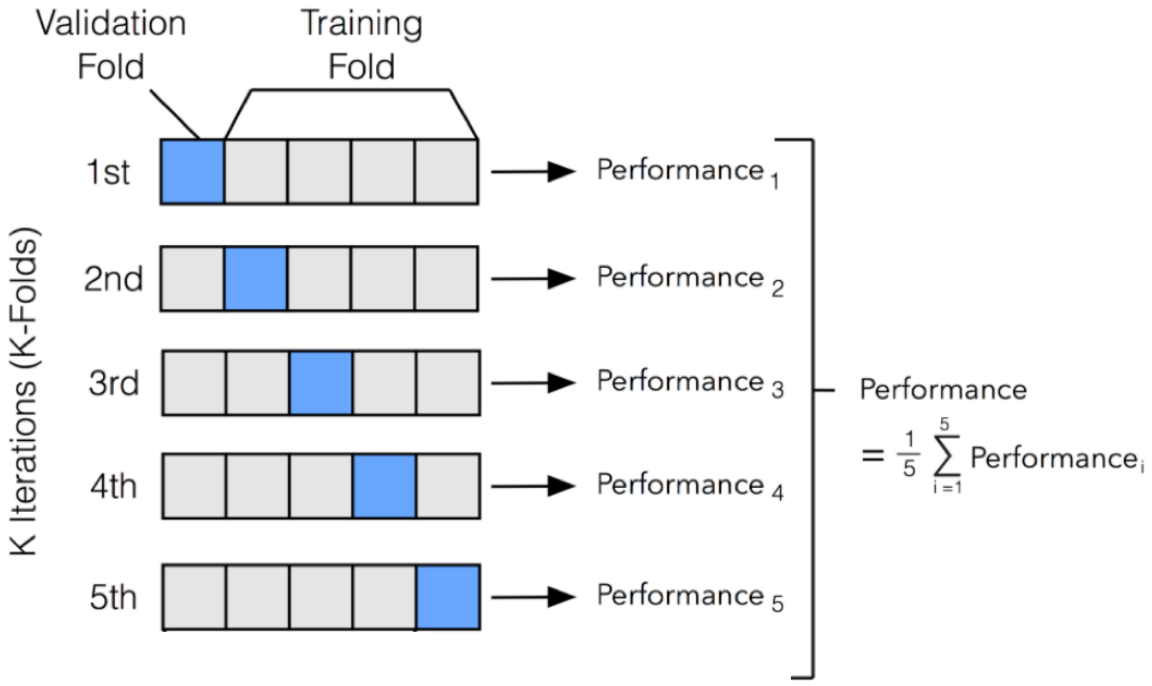
Exemplo: Overffiting.ipynb

Como detectar o sobreajuste



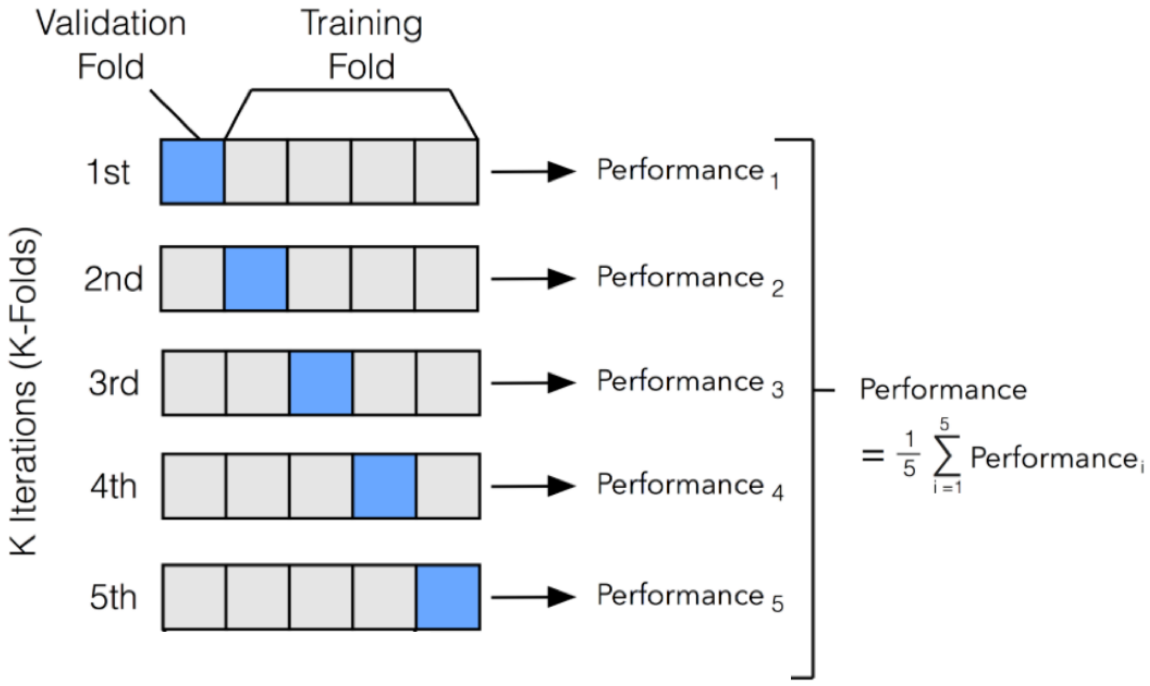
- **Conjunto de Validação:** No caso de ***modelos de aprendizado iterativo***, como redes neurais, podemos ***monitorar o desempenho do modelo em ambos os conjuntos*** durante o treinamento.
- Se o ***erro no conjunto de validação aumentar e no conjunto de treinamento diminuir*** ao longo das épocas de treinamento, isso é ***um claro sinal de sobreajuste***.
  - Como vimos antes, nesse caso, podemos usar a ***parada antecipada*** (*early stopping*).

Como detectar o sobreajuste



- Técnicas de ***validação cruzada avaliam o desempenho*** do modelo em ***várias divisões de dados de treinamento e validação***.
- Se o ***modelo mostrar variações significativas no desempenho*** (i.e., erro e variância) ***entre as divisões*** dos dados, isso pode ser um ***sinal de sobreajuste ou subajuste***.
- Técnicas de validação cruzada que podem ser usadas: *k-fold*, *leave-p-out*, *holdout*.
  - Na prática, o *k*-fold é a mais usada.

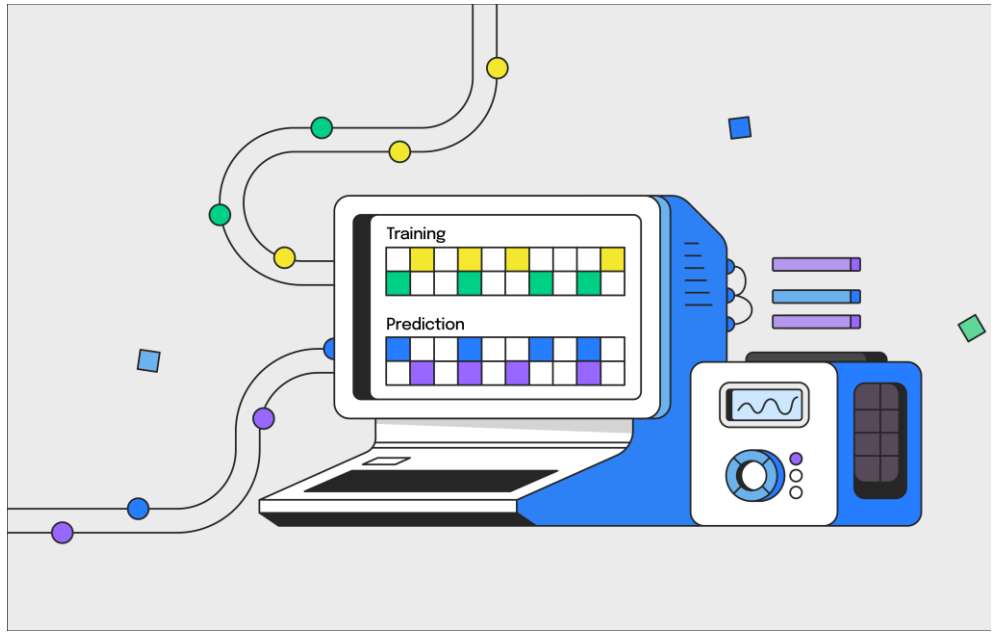
Como detectar o sobreajuste: *k-fold*



- A técnica ***divide o conjunto de dados em k subconjuntos***, chamados dobras (*folds*), ***treinando*** e ***avaliando o modelo k vezes*** (e.g., MSE ou acurácia), cada vez com um subconjunto diferente como conjunto de teste e os k-1 subconjuntos restantes como conjunto de treinamento.
- Ao final, ***calcula-se a média e a variância das k métricas*** individuais de desempenho para fornecer uma única ***métrica geral do modelo***.

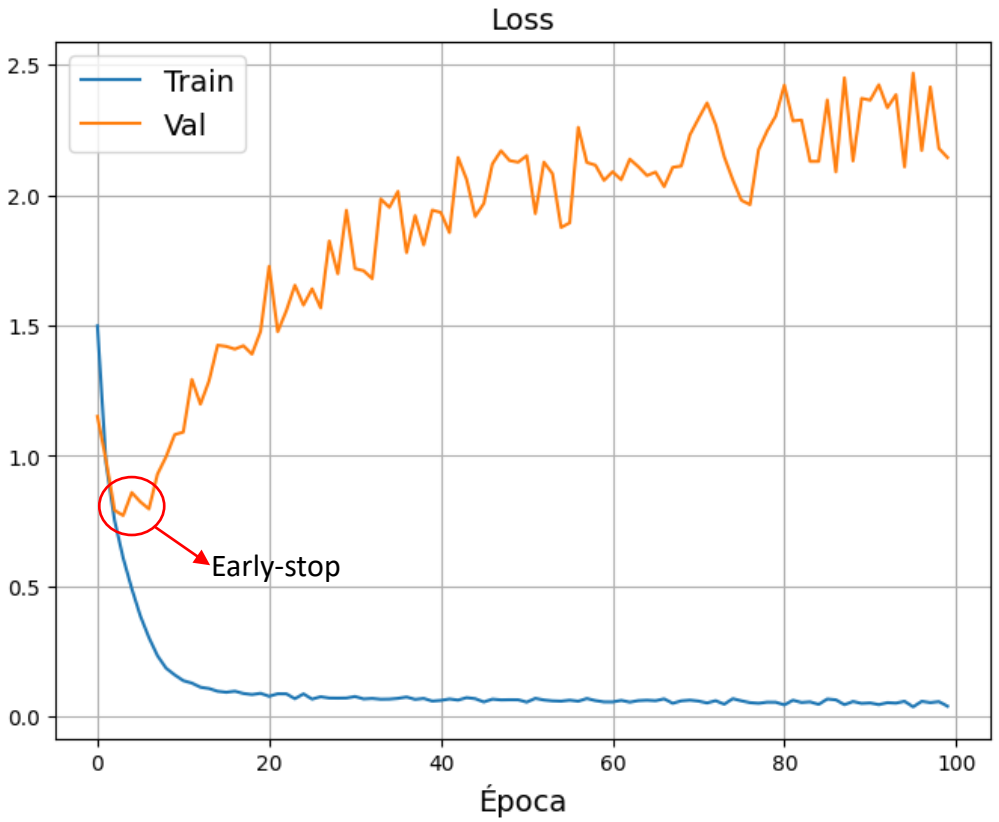
Exemplo: validacao_cruzada_k_fold.ipynb

Como evitar o sobreajuste?



- **Coletar mais dados** de treinamento é uma das ***estratégias mais eficazes*** para ***melhorar a capacidade de generalização*** do modelo, consequentemente, reduzindo o sobreajuste.
- Porém, em algumas situações, a ***coleta de dados*** adicionais pode ser muito ***cara***, ***demorada*** ou ***simplesmente impossível***.
  - Por coleta, entende-se que a ***rotulagem também está inclusa***.

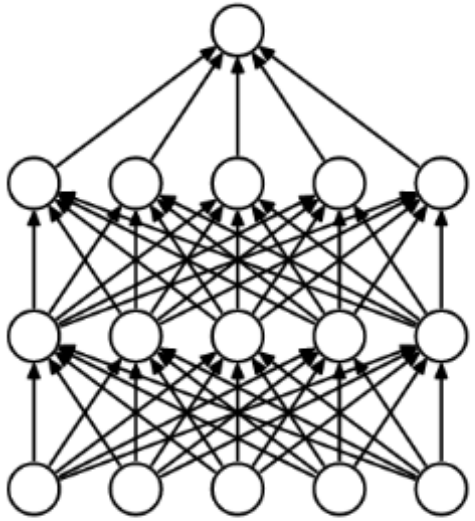
Como evitar o sobreajuste?



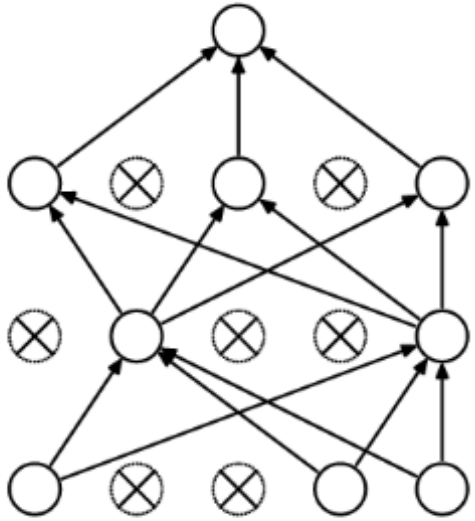
- **Regularização:** Utilizar técnicas de regularização, como L1, L2 ou *Early stopping* e *Dropout* em redes neurais, para reduzir a complexidade do modelo.
- ***Early stopping*** (ou parada antecipada) é uma técnica de ***regularização temporal***.
- A ideia é ***interromper o treinamento do modelo assim que o desempenho no conjunto de validação começa a piorar***, em vez de continuar até que o modelo se ajuste demais aos dados de treinamento.
  - ***Vantagens***: economiza tempo e recursos computacionais.

Exemplo: Sobreajuste_e_early_stop.ipynb

Como evitar o sobreajuste?



(a) Standard Neural Net

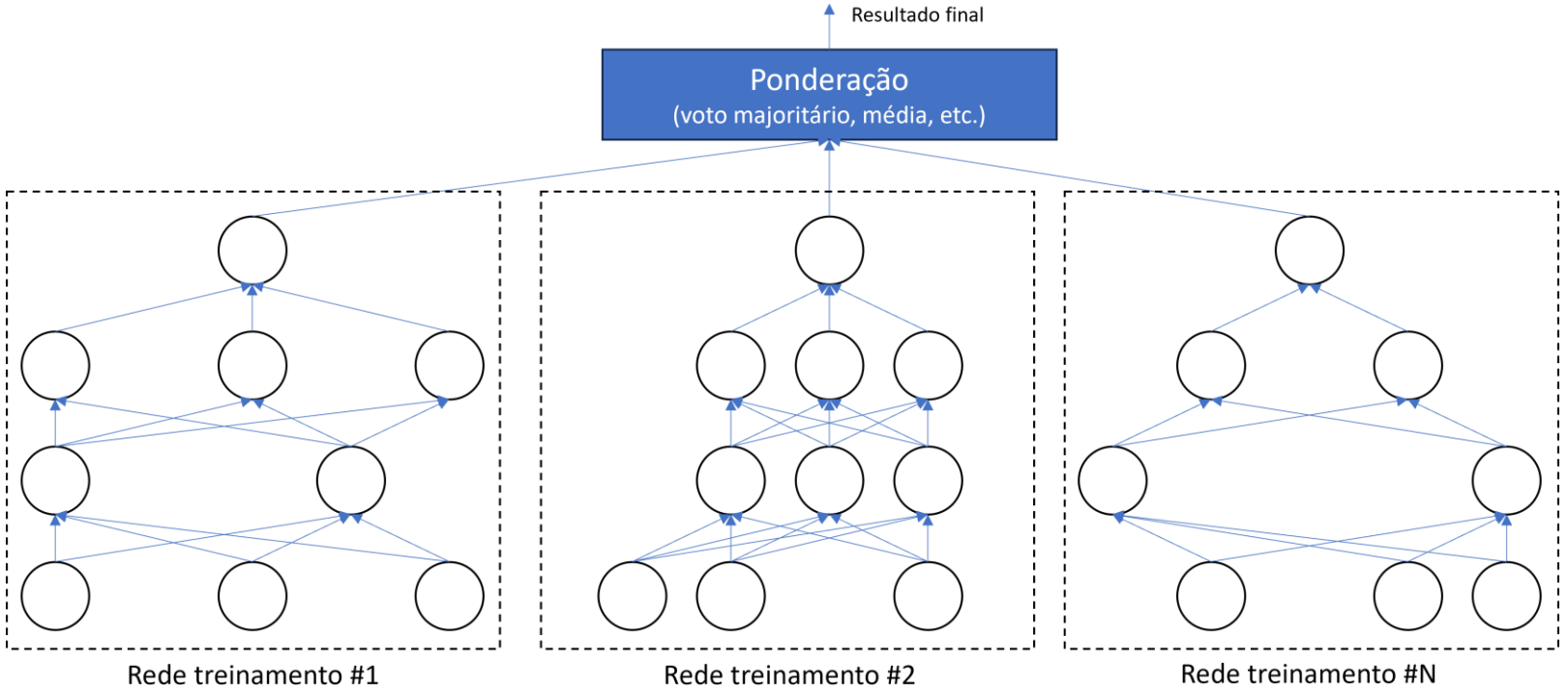


(b) After applying dropout.

- O ***dropout*** envolve aleatoriamente ***"desligar" ou "descartar" um subconjunto dos neurônios em cada camada***, ***durante o treinamento*** da rede neural.
- Isso significa que esses neurônios não contribuirão para o cálculo das saídas da rede durante essa iteração de treinamento.
- Desta forma, o ***modelo*** fica ***menos complexo***, ***diminuindo*** as chances de ***sobreajuste***.

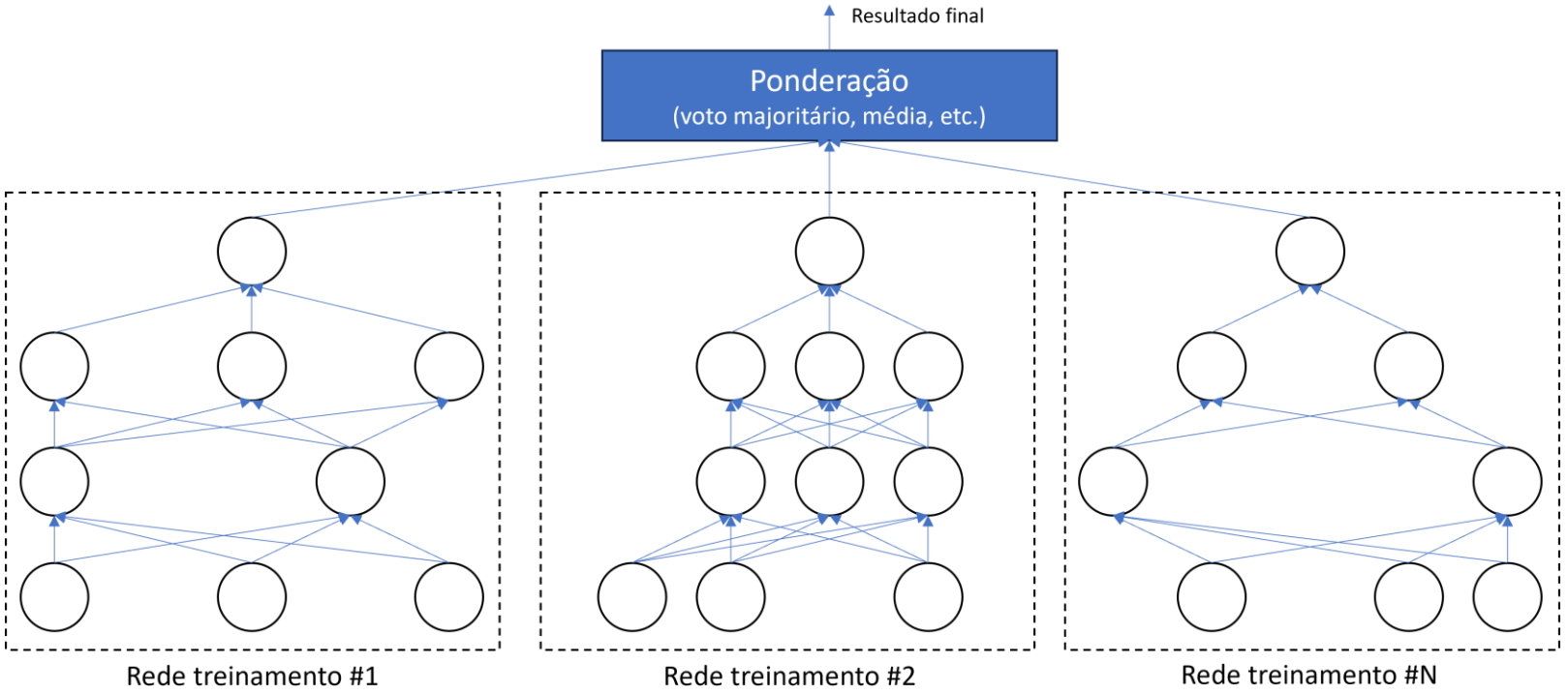
Exemplo: Sobreajuste_e_dropout.ipynb

Como evitar o sobreajuste?



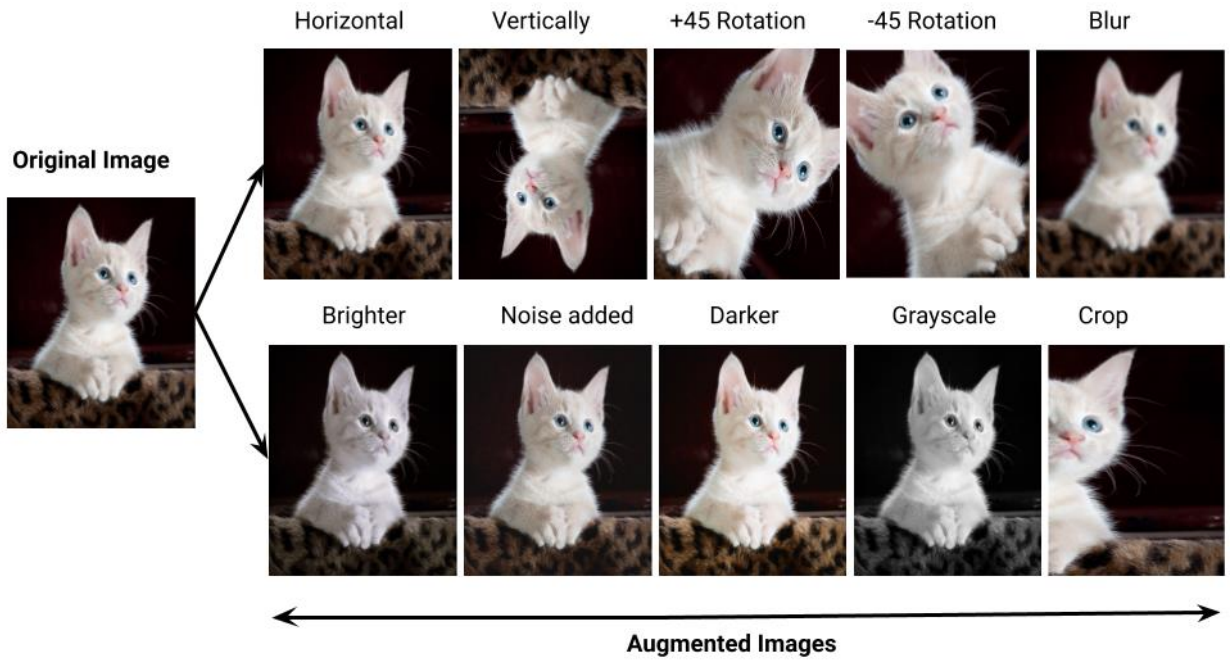
- ***Heuristicamente***, quando ***desligamos*** diferentes conjuntos de ***neurônios***, é como se ***estivéssemos treinando várias redes*** neurais diferentes.

Como evitar o sobreajuste?



- Durante a ***inferência***, os ***resultados dessas redes são ponderados*** (e.g., média ou voto majoritário) para produzir a saída final da rede.
- As ***diferentes redes se adaptarão de diferentes maneiras***, e assim, esperançosamente, o efeito do *dropout* será ***reduzir o sobreajuste***.

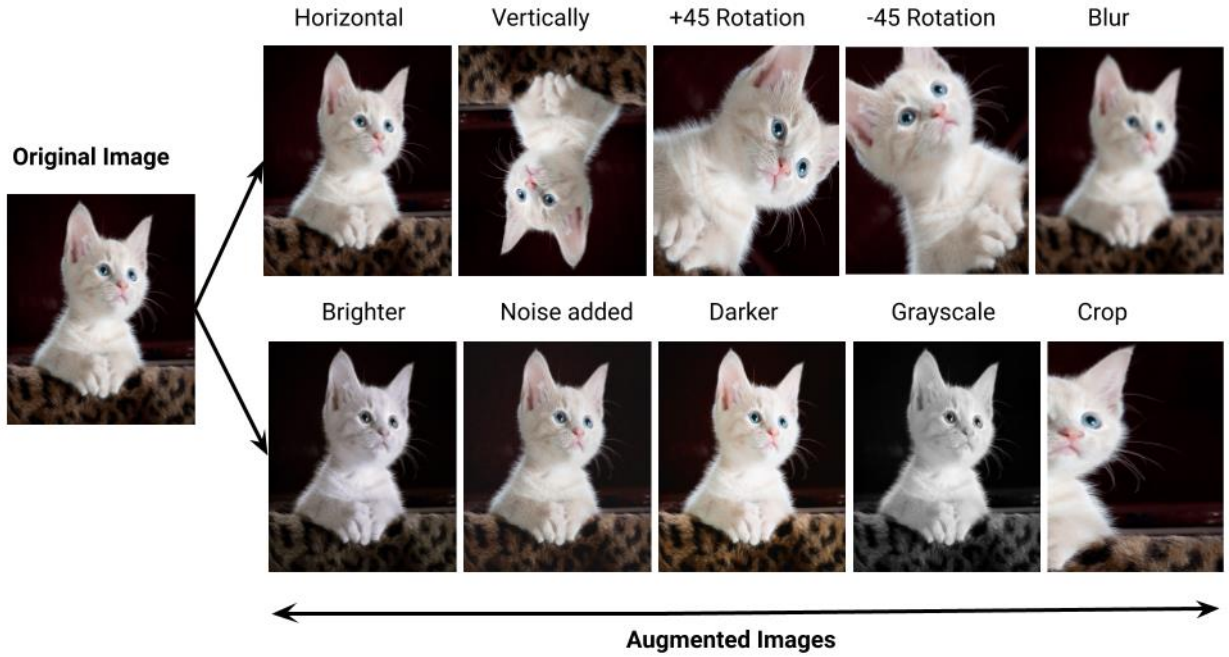
Como evitar o sobreajuste?



- **Aumentar artificialmente os dados (*Data Augmentation*):** O objetivo é ***aumentar*** a ***variedade*** e a ***diversidade*** dos dados de treinamento ***criando novos exemplos a partir dos dados existentes***.
- Essa técnica ajuda a melhorar o desempenho do modelo, tornando-o mais ***robusto*** e capaz de ***generalizar melhor***.

Exemplo: Treinamento_com_e_sem_data_augmentation.ipynb

Como evitar o sobreajuste?



- Quando o problema envolve ***imagens***, podemos aplicar ***transformações***, como ***rotação***, ***espelhamento*** e ***corte***, modificação do ***brilho***, adição de ***ruído***, ***desfoque***, etc.
- O ***data augmentation torna CNNs robustas à rotação das imagens***, pois a rede é treinada com a ***mesma imagem com diferentes rotações***.
- Assim a rede aprende a ***reconhecer padrões independentemente da orientação***.

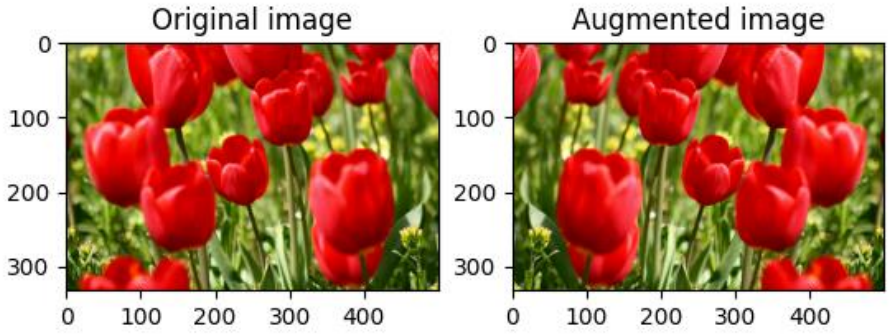
Data augmentation com Tensorflow

```
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)
```

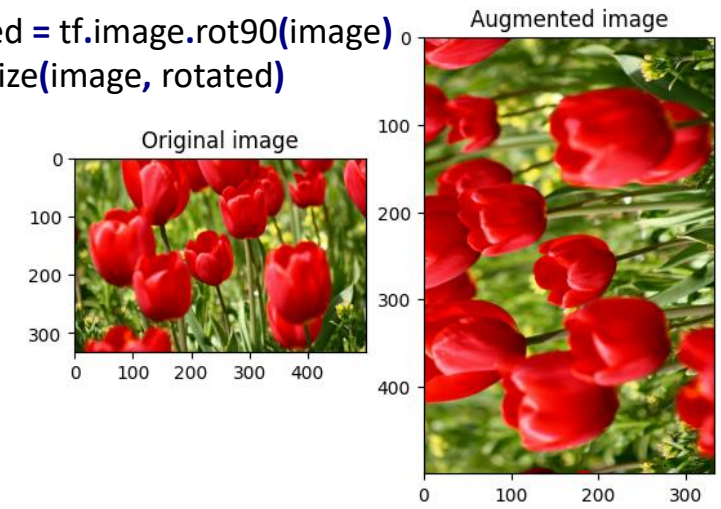
- O Tensorflow possui várias ***camadas*** que podem ser usadas para criar novas imagens a partir das existentes:
  - tf.keras.layers.RandomCrop(),
  - tf.keras.layers.RandomZoom(),
  - tf.keras.layers.RandomFlip(),
  - tf.keras.layers.RandomRotation(), etc.
- Essas camadas podem ser ***adicionadas como camadas iniciais em seu modelo*** e o TensorFlow cuidará automaticamente do pré-processamento durante o treinamento e as desligará durante a inferência.

Data augmentation com Tensorflow

```
flipped = tf.image.flip_left_right(image)
visualize(image, flipped)
```



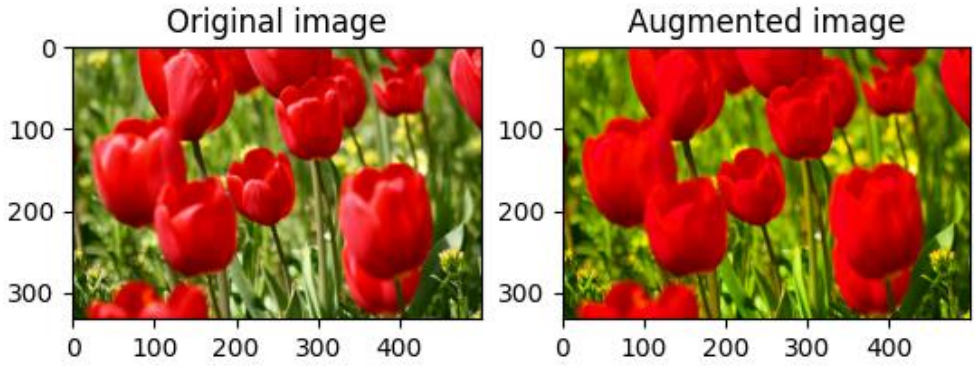

```
rotated = tf.image.rot90(image)
visualize(image, rotated)
```



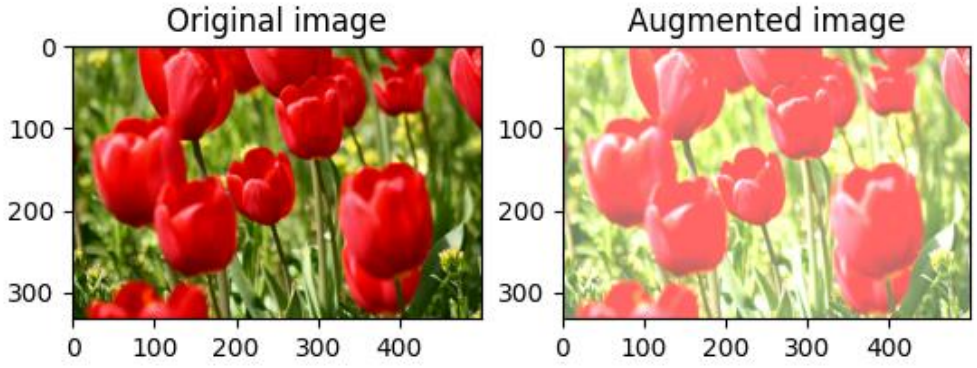
- Além das camadas, o Tensorflow possui um módulo de pré-processamento de imagens chamado ***tf.image*** que contém ***funções*** que também podem ser usadas para criar novas imagens.
- Elas são usadas para um controle mais preciso do pré-processamento.

Data augmentation com Tensorflow

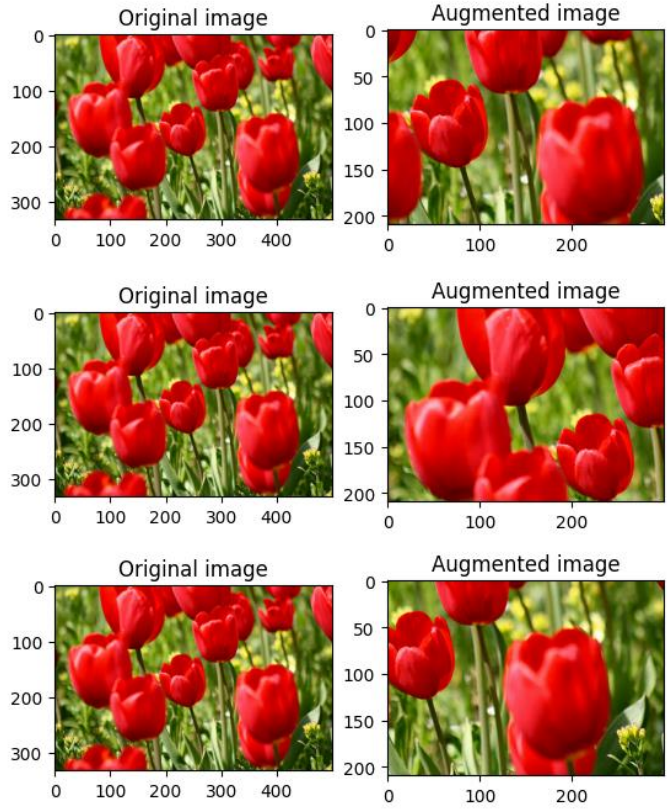
```
saturated = tf.image.adjust_saturation(image, 3)
visualize(image, saturated)
```




```
bright = tf.image.adjust_brightness(image, 0.4)
visualize(image, bright)
```

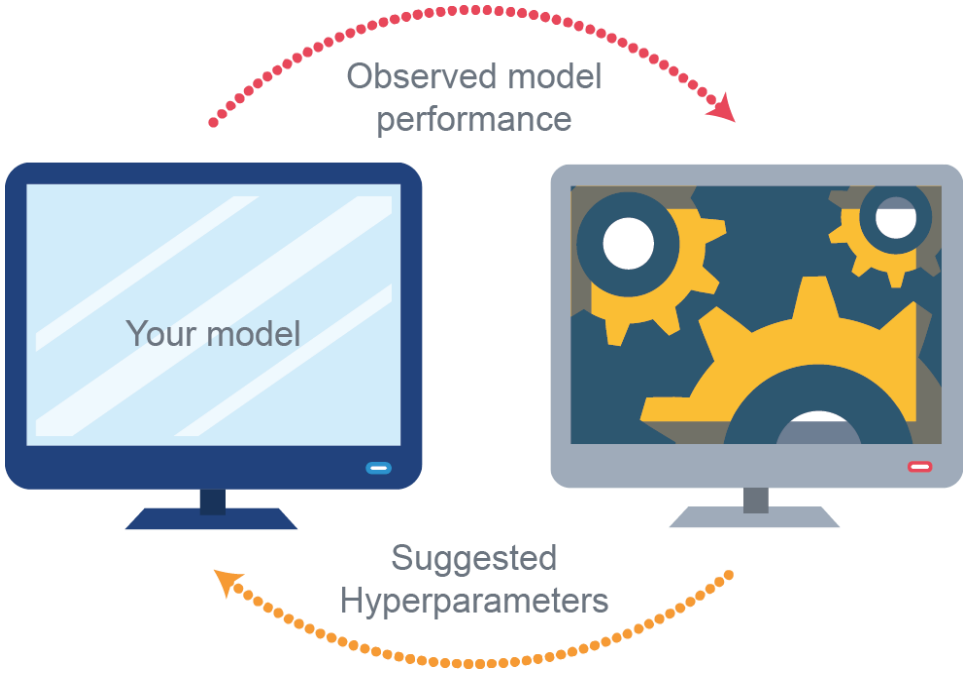



```
for i in range(3):
  seed = (i, 0)  # tuple of size (2,)
  stateless_random_crop = tf.image.stateless_random_crop(
     image, size=[210, 300, 3], seed=seed)
  visualize(image, stateless_random_crop)
```



Exemplo: data_augmentation.ipynb

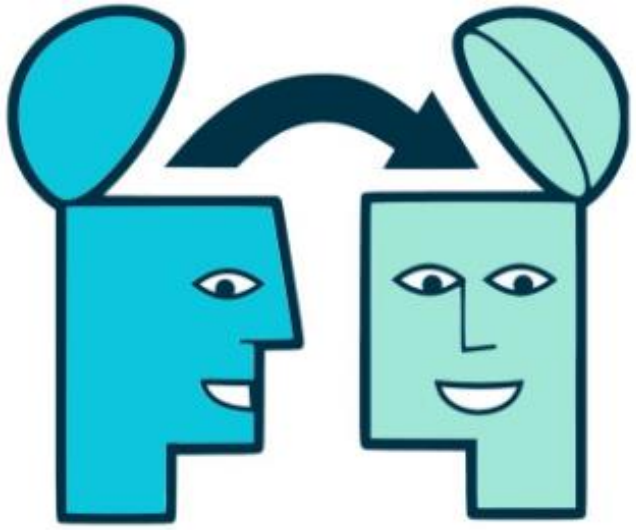
Como evitar o sobreajuste?



- Um ***modelo excessivamente complexo*** tem grandes chances de se ***sobreajustar***.
- Portanto, ***modelos menos complexos*** (i.e., mais simples), com menos parâmetros, ***tendem a não se sobreajustar e a generalizar melhor***.
- O objetivo é encontrar um modelo com ***complexidade suficiente para capturar o padrão geral por trás dos dados***.
- ***Otimização hiperparamétrica*** pode ajustar a complexidade do modelo e ***encontrar a configuração que oferece um bom equilíbrio entre viés e variância***.

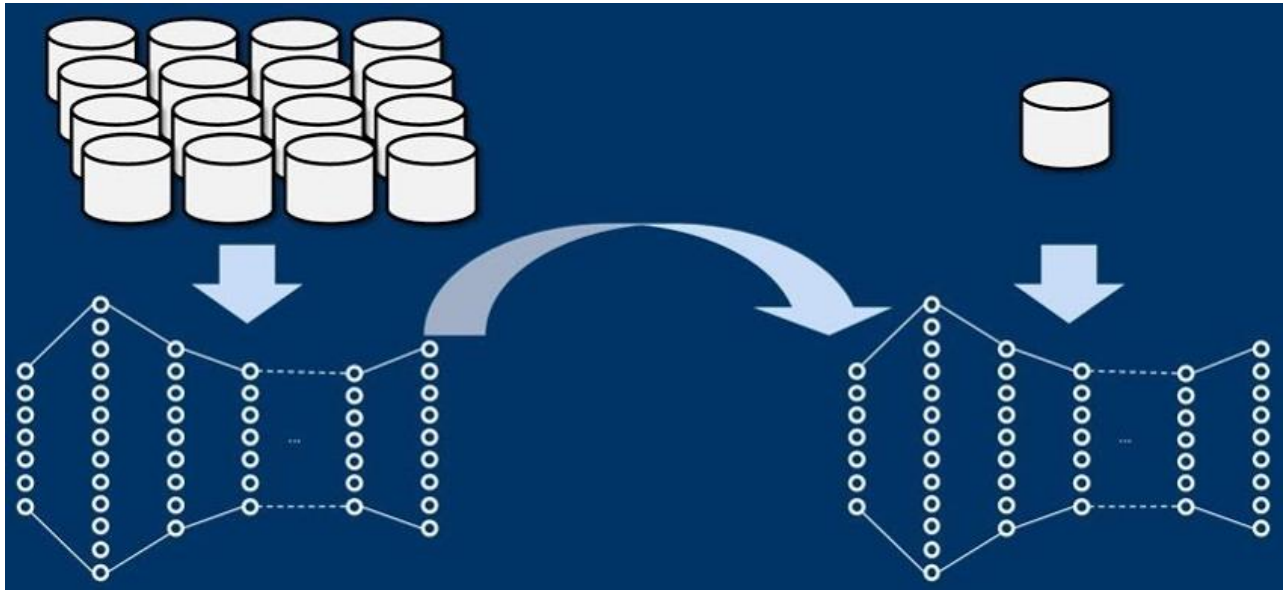
Exemplo: Sobreajuste_e_otimização_hiperparamétrica.ipynb

Como evitar o sobreajuste?



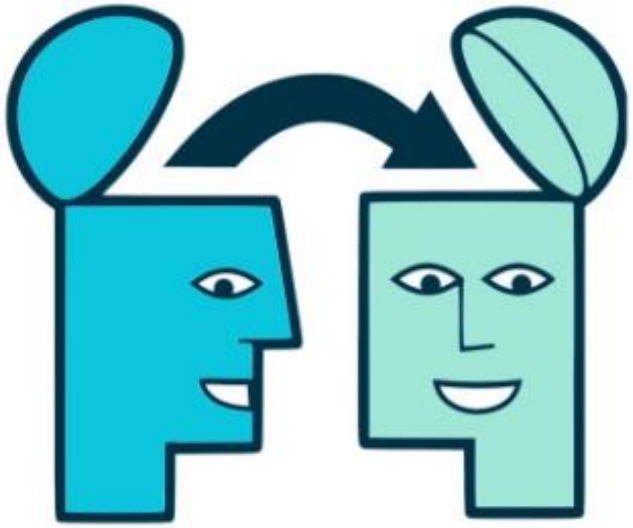
- ***Transfer learning*** ou ***transferência de aprendizado*** é uma abordagem que envolve o uso de um ***modelo pré-treinado em uma tarefa*** (ou domínio) relacionada ***como ponto de partida para uma nova tarefa***.

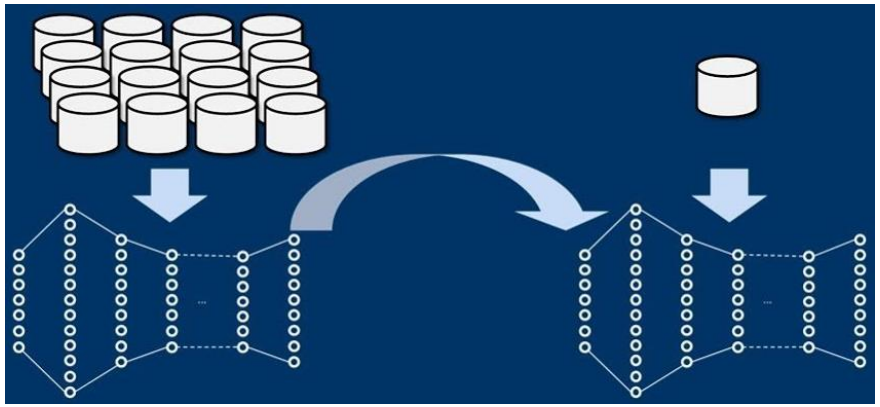
Como evitar o sobreajuste?



- Usamos o ***conhecimento adquirido*** por um ***modelo treinando em uma grande base de dados*** como ***ponto de partida*** para treinar um modelo com uma ***base de treinamento menor***, ***mas que seja relaciona à primeira***.

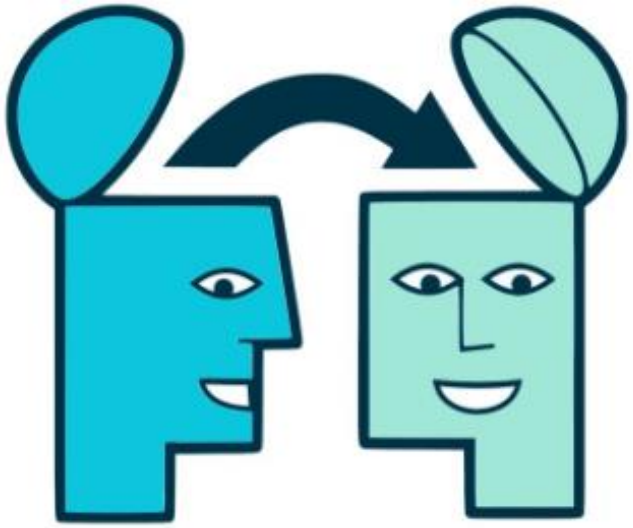
Transfer learning

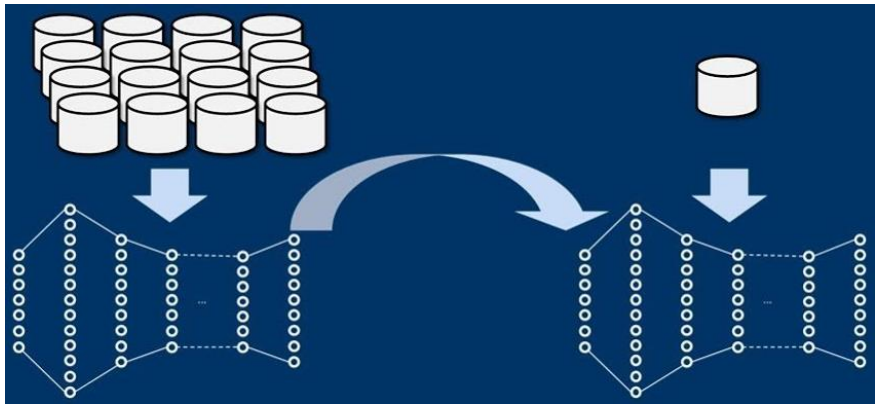




- Suponham que temos uma ***rede pré-treinada*** para classificar imagens em 100 categorias diferentes, incluindo ***animais***, ***plantas***, ***veículos***, etc. e queremos treinar uma ***nova rede para classificar tipos específicos de veículos***.
- Por essas tarefas serem muito semelhantes, ***podemos reutilizar partes da primeira rede***.

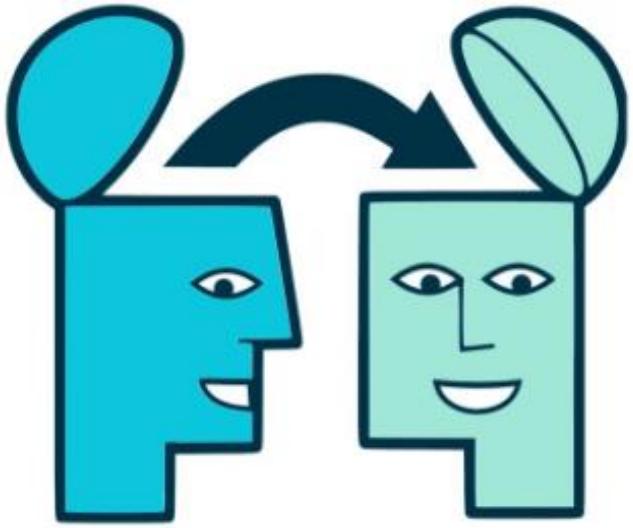
Como evitar o sobreajuste?

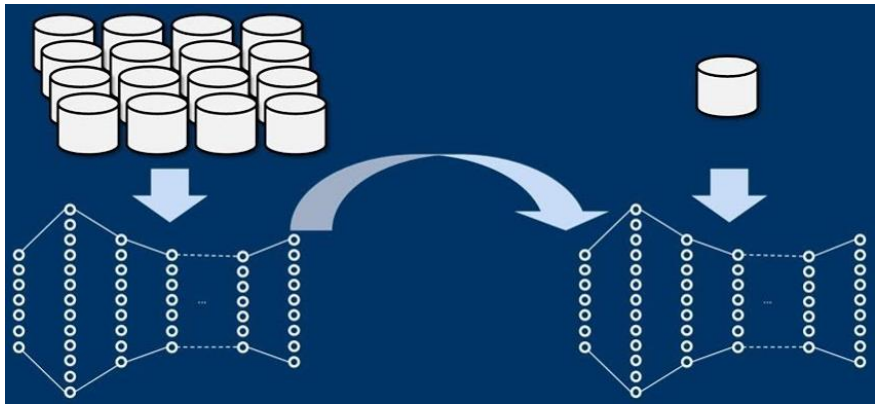




- Modelos pré-treinados em grandes conjuntos de dados ***extraem características relevantes dos dados***.
- Essas ***características são transferidas para o novo modelo***, que pode se beneficiar de uma ***inicialização mais informada***.

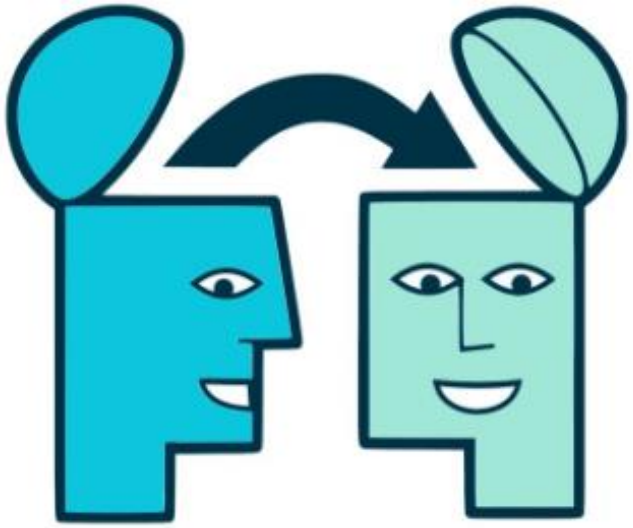
Transfer learning

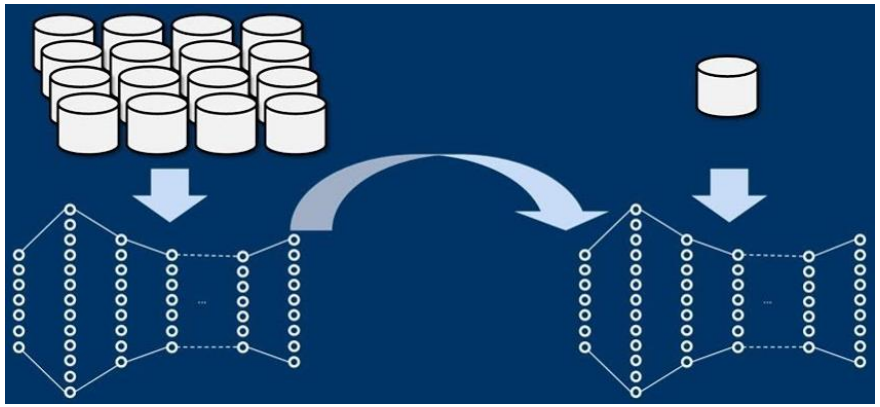




- O *transfer learning* ***pode evitar*** que o ***novo modelo sobreajuste***, pois ele ***já começa com características relevantes***.
- Como o ***modelo pré-treinado já adquiriu um bom entendimento de características genéricas*** em uma tarefa relacionada, o ***novo modelo pode exigir menos dados de treinamento*** para alcançar um bom desempenho.

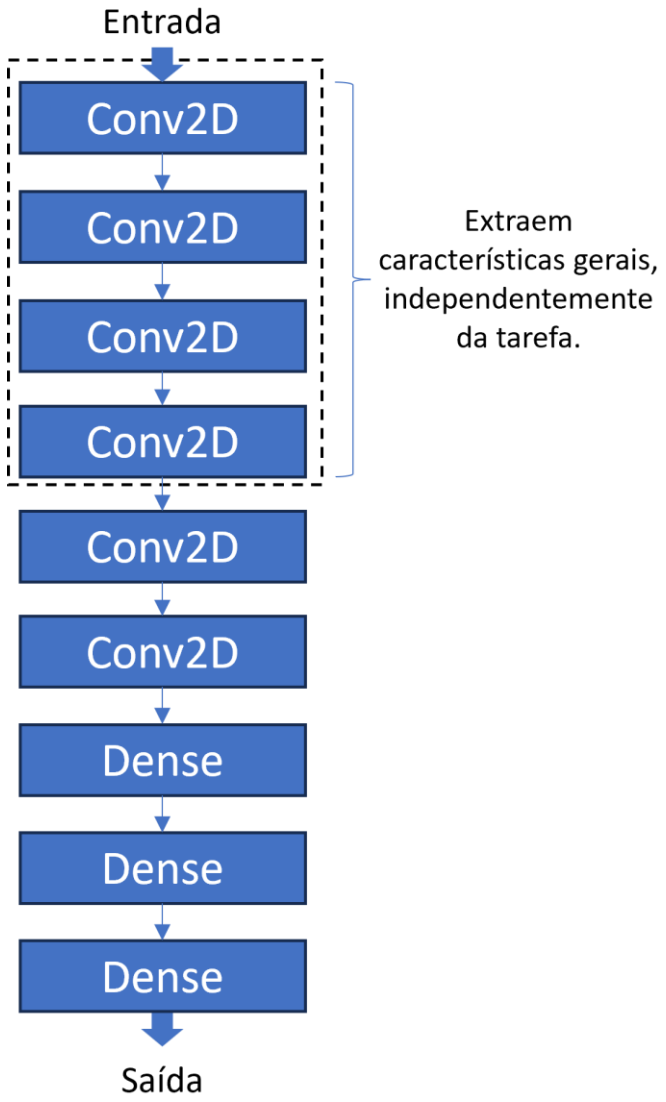
Transfer learning





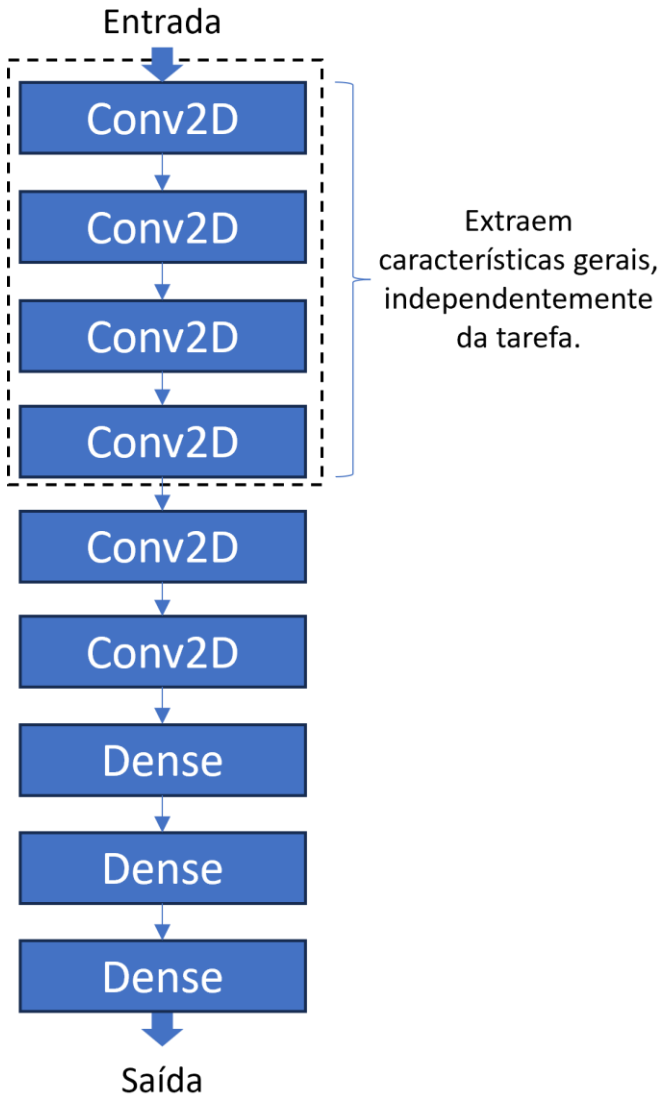
- O *transfer learning* é particularmente ***útil quando os dados de treinamento são escassos***, o que é uma situação ***propensa ao sobreajuste***.
- Além de requerer menos dados, o *transfer learning* ***acelera consideravelmente o treinamento***.

Rede convolucional



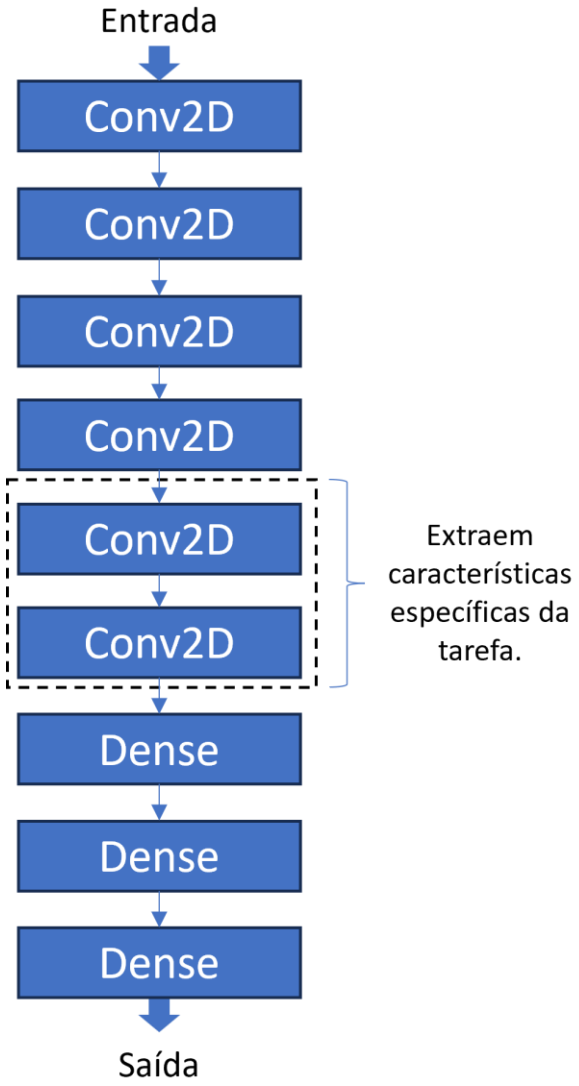
- Para entender como o *transfer learning*, funciona, vamos usar uma ***rede convolucional*** como exemplo.
- Nestas redes, as ***camadas convolucionais iniciais***, ***aprendem*** a ***detectar características simples*** (i.e., de baixo nível), como bordas (verticais e horizontais), linhas, cantos e texturas.
  - Características gerais, independentes da tarefa.

Rede convolucional



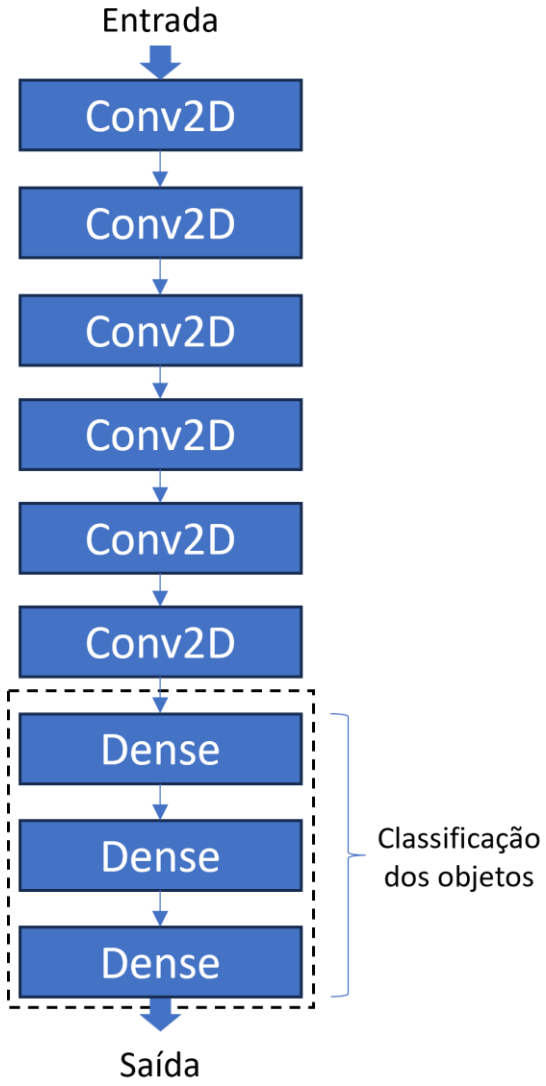
- As ***camadas intermediárias***, ***baseando-se nas características aprendidas pelas camadas anteriores***, ***aprendem a detectar características mais complexas***, como formas geométricas mais elaboradas, partes de objetos e texturas mais abstratas.
- Estas camadas capturaram características parciais de objetos, como olhos, narizes e partes do corpo.

Rede convolucional



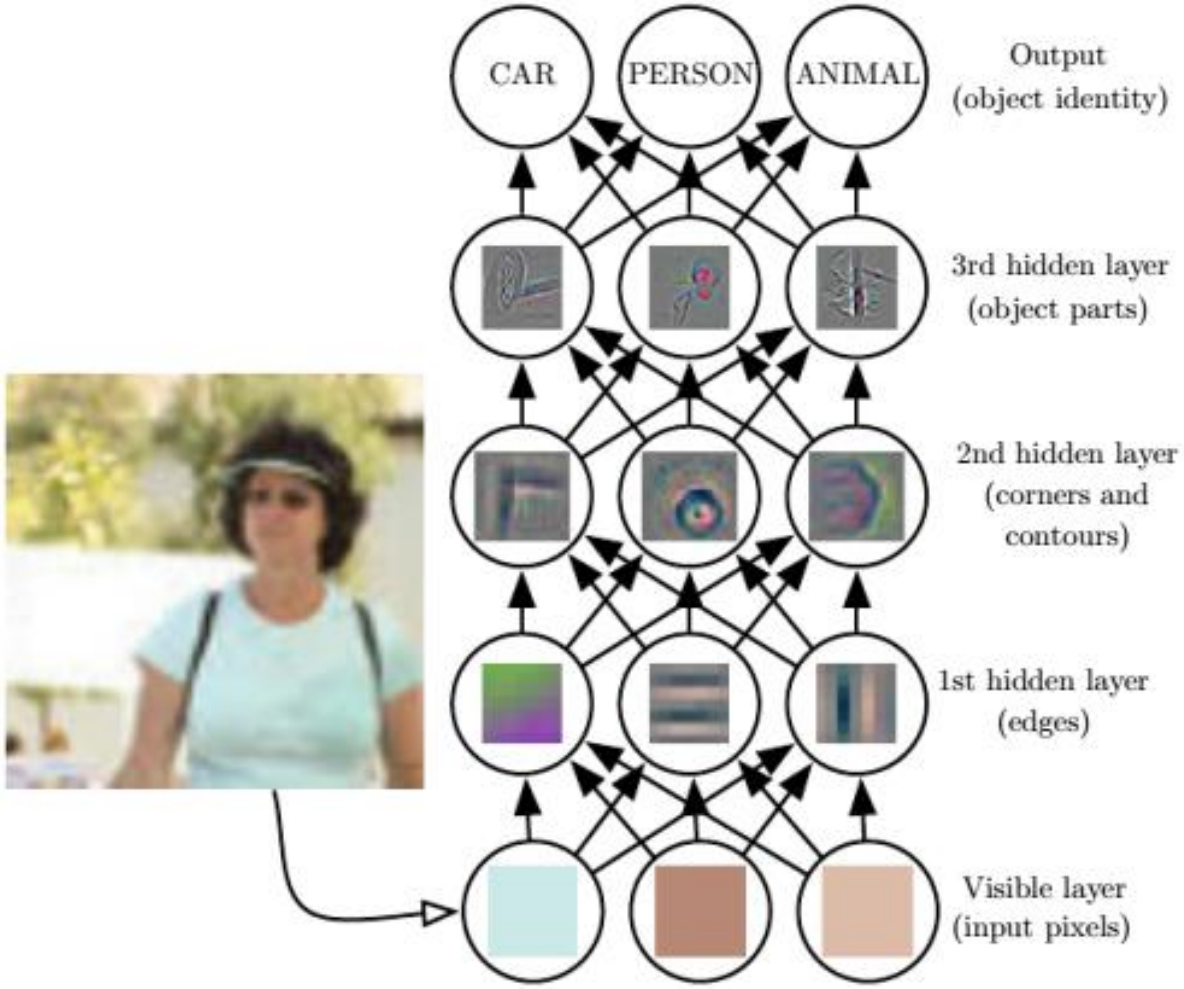
- Nas ***camadas mais profundas, combinando as características detectadas pelas camadas anteriores***, a ***rede convolucional tende a aprender características de alto nível***, como partes completas de objetos, objetos inteiros e até mesmo conceitos semânticos mais abstratos, como "rosto humano" ou "carro".

Rede convolucional



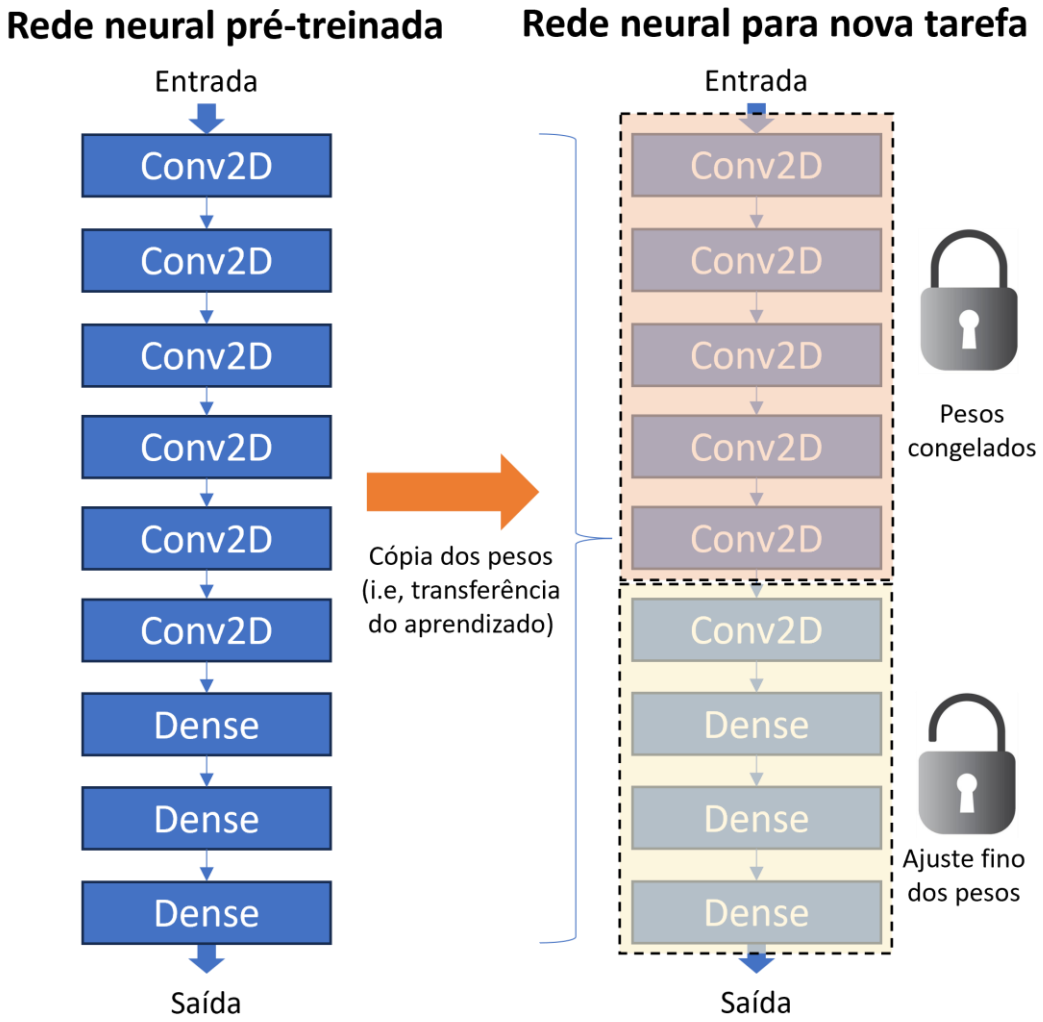
- As ***camadas densas recebem as características extraídas*** pelas camadas convolucionais e ***combinam essas informações*** para ***aprender padrões complexos*** e realizar a tarefa de ***classificação*** ou ***regressão***.
- Em tarefas de classificação, elas mapeiam as características para as classes alvo, enquanto em tarefas de regressão, produzem uma saída contínua.

Rede convolucional



- ***Combinando conceitos simples***, como cantos e contornos, ***com conceitos mais complexos***, como partes de um objeto, ***uma rede convolucional aprende a detectar pessoas***, por exemplo.

Transfer learning



- Portanto, o que fazemos no *transfer learning* é ***congelar*** os pesos das camadas iniciais (i.e., ***características gerais, independentes da tarefa***) e apenas aplicar ***atualizações*** (i.e., gradiente descendente) ***aos pesos das camadas finais***.
- Se a tarefa da rede neural pré-treinada for ***muito similar à nova tarefa***, pode-se ***treinar apenas as camadas densas***.

Transfer learning

- Em geral, ***começamos com todas as camadas congeladas***, com ***exceção da camada densa de saída*** (i.e., a camada com função de ativação *softmax*).
- ***Treinamos*** o modelo por um número de épocas pequeno e ***verificamos seu desempenho***.
- Na sequência, ***descongelamos uma ou duas camadas*** ocultas do topo da rede e ***treinamos*** o modelo para ver se seu ***desempenho melhora***.
- Esse ***processo*** pode ser ***repetido até que não se obtenha mais melhorias*** no desempenho.
- ***Quanto mais dados de treinamento*** tivermos, ***mais camadas podemos descongelar***.
- Também é útil ***reduzir a taxa de aprendizado ao descongelar camadas***, pois isso evitará destruir os pesos já ajustados.

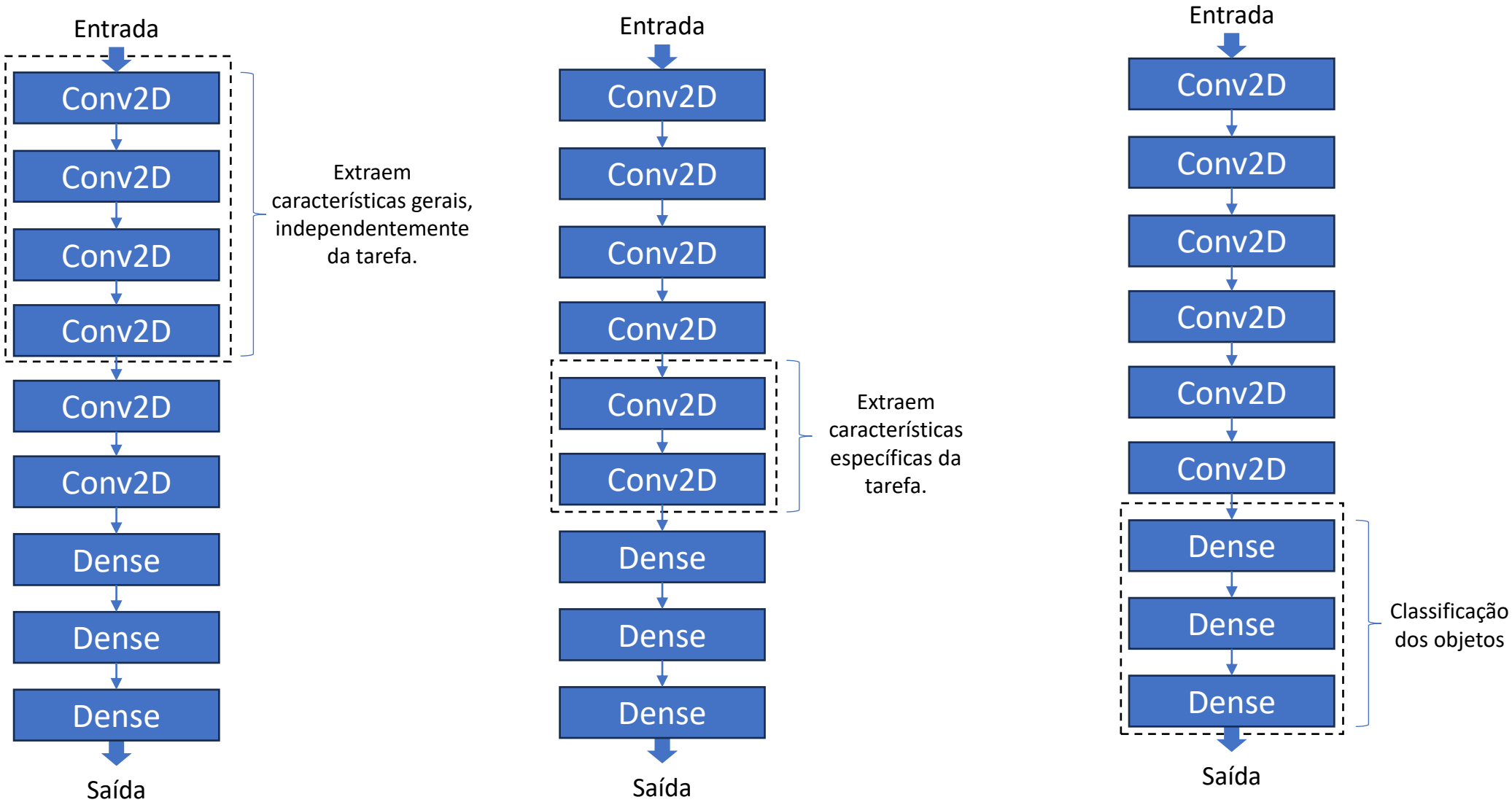
Exemplo: transfer_learning.ipynb

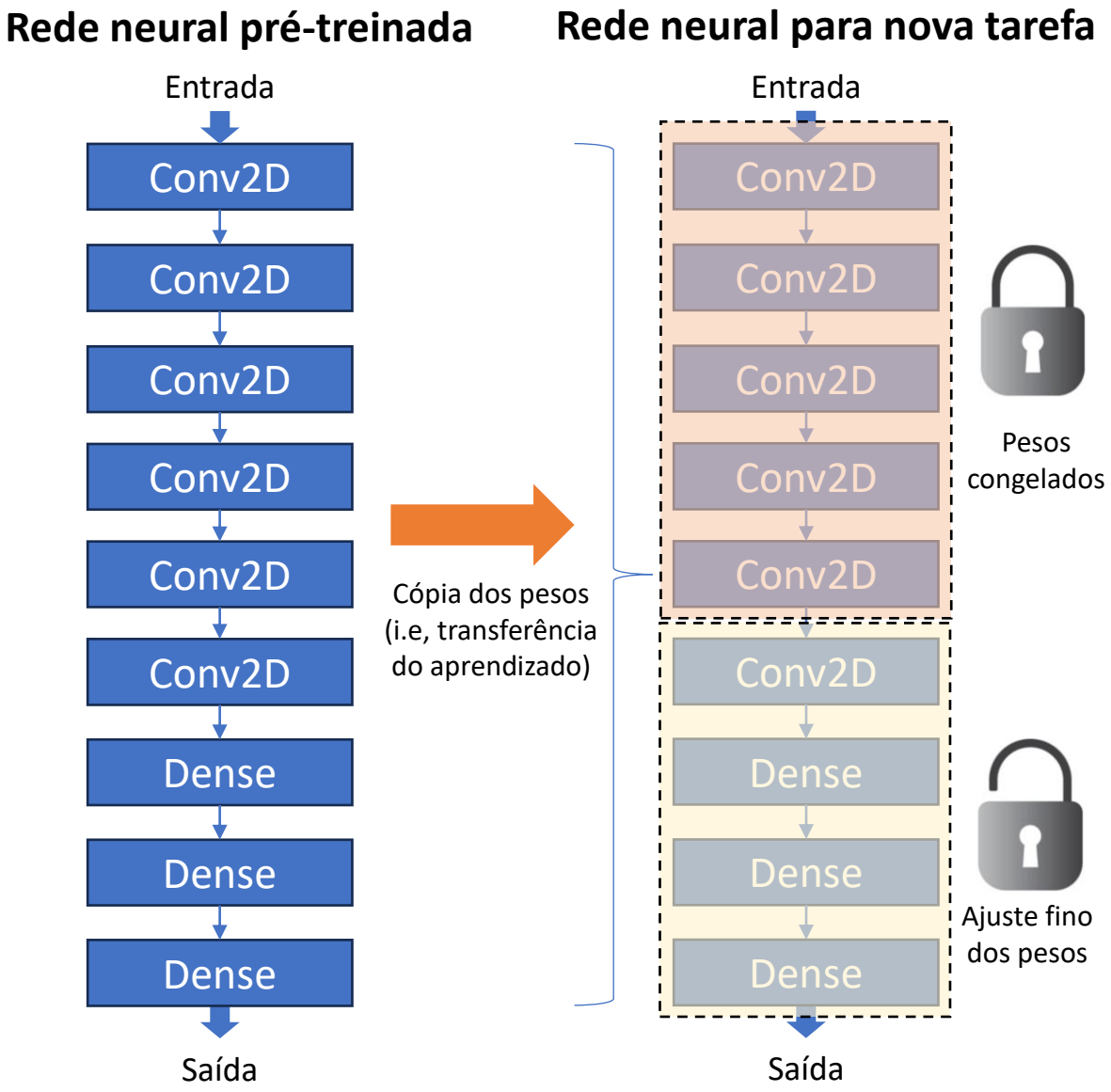
Atividades

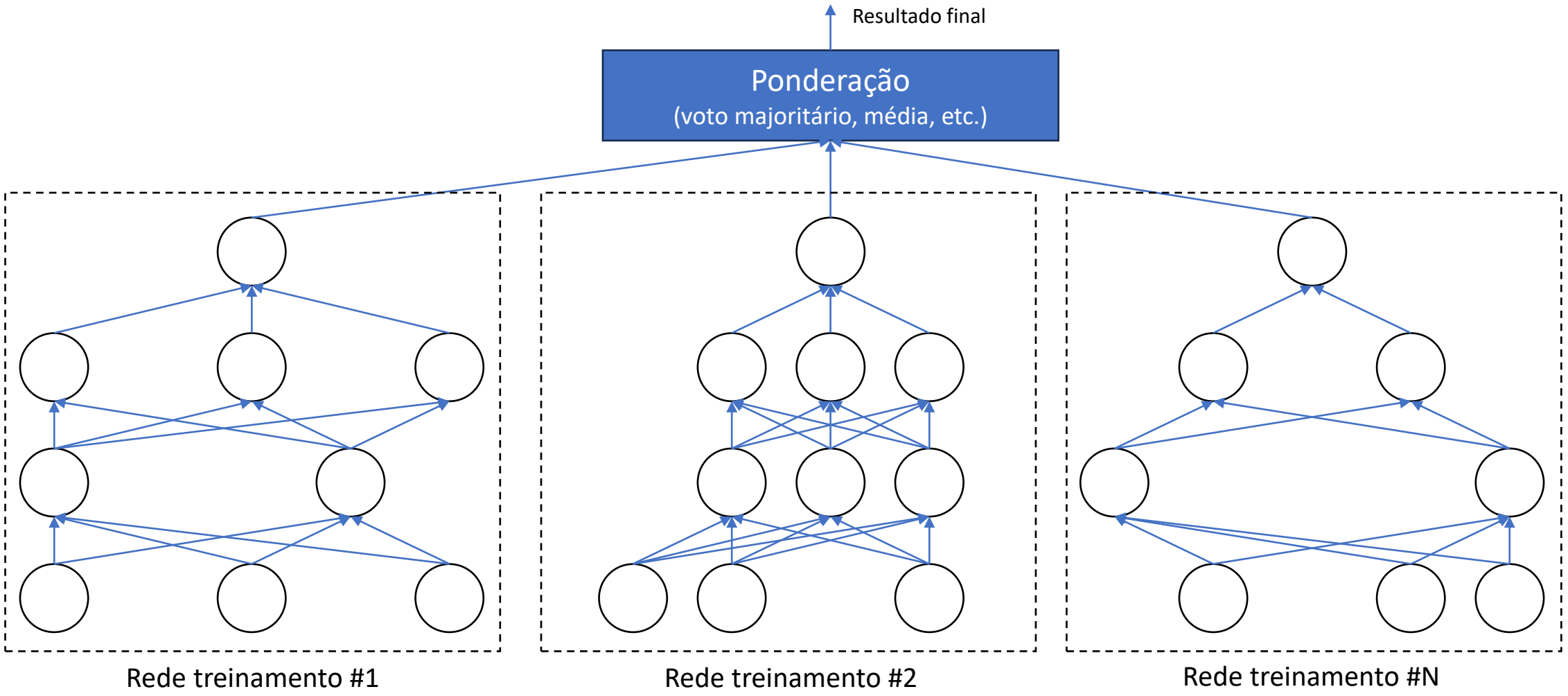
- Quiz: “***TP557 – Evitando o sobreajuste***”.
- Exercício: Transfer learning.

Perguntas?

Obrigado!


Entrada
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Dense
Dense
Dense
Saída
Extraem características gerais, independentemente da tarefa.
Entrada
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Dense
Dense
Dense
Saída
Extraem características específicas da tarefa.
Entrada
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Dense
Dense
Dense
Saída
Classificação dos objetos


Rede neural pré-treinada
Entrada
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Dense
Dense
Dense
Saída
Cópia dos pesos
(i.e, transferência
do aprendizado)
Rede neural para nova tarefa
Entrada
Conv2D
Conv2D
Conv2D
Conv2D
Conv2D
Pesos
congelados
Conv2D
Dense
Dense
Dense
Ajuste fino
dos pesos
Saída


Resultado final
Ponderação
(voto majoritário, média, etc.)
Rede treinamento #1
Rede treinamento #2
Rede treinamento #N