



# FIAP

# ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

## DISRUPTIVE ARCHITECTURES: IOT, IOB & IA

### 09 – Redes Neurais Transfer Learning e outras técnicas



Prof. Airton Y. C. Toyofuku



[profairton.toyofuku@fiap.com.br](mailto:profairton.toyofuku@fiap.com.br)

# 1º Validation

## Processo de validação



- ☐ Processo usado durante o treinamento do modelo
- ☐ Avalia o desempenho do modelo em um conjunto de dados separado
- ☐ Conjunto de validação é uma porção do conjunto de dados total
- ☐ Não é usado para treinar o modelo, mas para avaliar sua performance durante o treinamento

## Conjuntos de dados em redes neurais



- ☐ Conjunto de treinamento: usado para ajustar os pesos e bias do modelo
- ☐ Conjunto de validação: usado para monitorar a performance do modelo durante o treinamento
- ☐ Conjunto de teste: usado para avaliar a performance do modelo após o treinamento

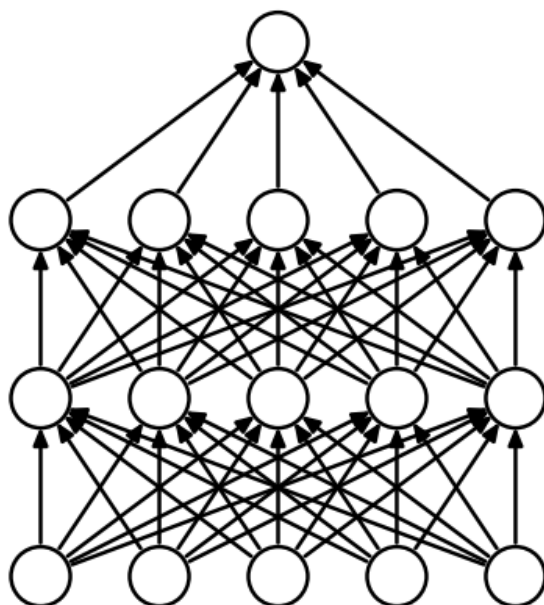
## Importância da validação



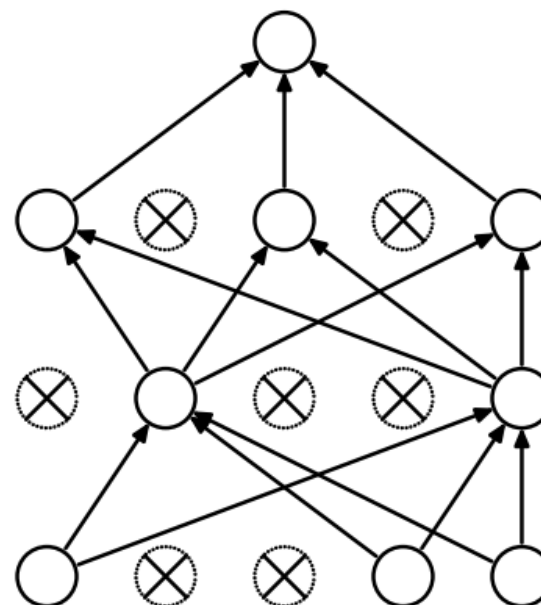
- ☐ Avaliar a capacidade do modelo de generalizar os padrões aprendidos durante o treinamento
- ☐ Ajustar hiperparâmetros do modelo com base no desempenho no conjunto de validação
- ☐ Evitar o overfitting, onde o modelo se ajusta excessivamente aos dados de treinamento

Processo de Validação  $\neq$  Processo de Teste!!!!

# 2º Dropout



(a) Standard Neural Net



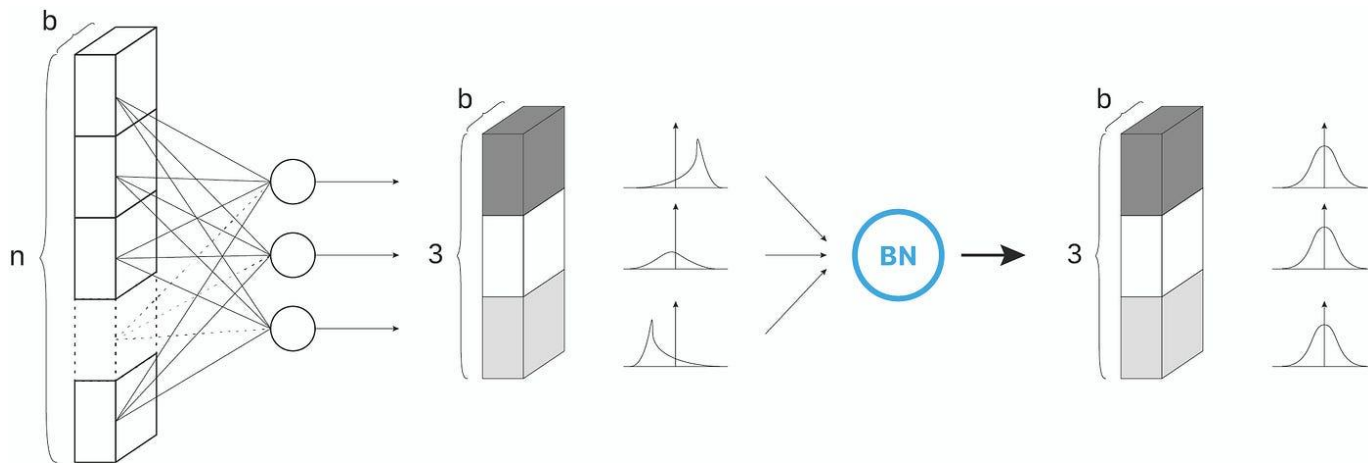
(b) After applying dropout.

Fonte: Srivastava et al. in [Dropout: A Simple Way to Prevent Neural Networks from Overfitting](#)

- ❖ Consiste em, aleatoriamente, ignorar alguns neurônios durante a etapa de treinamento para prevenir que a Rede Neural se especialize demais nos dados, evitando Overfitting;
- ❖ Nem sempre da certo...

# 3º Batch Normalization

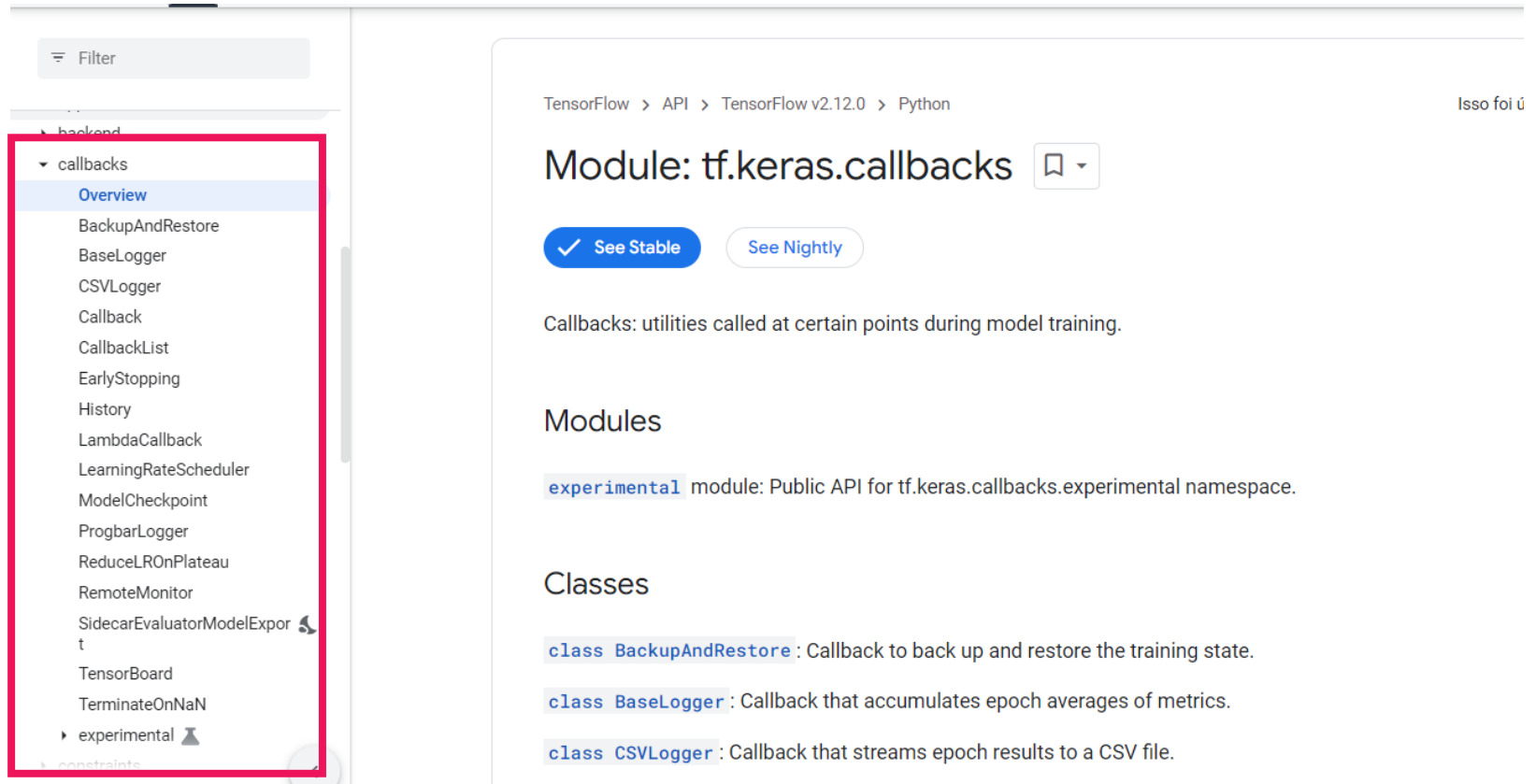
- ❖ Técnica usada em redes neurais para normalizar as ativações intermediárias durante o treinamento;
- ❖ A ideia é ajustar as ativações intermediárias em cada camada da rede para ter uma média zero e uma variância unitária em relação aos valores das ativações;
- ❖ É realizada adicionando uma camada de normalização em lote após as camadas de convolução ou após as camadas densas totalmente conectadas;
- ❖ Apresenta as seguintes vantagens:
  - ❖ **Aceleração do treinamento:** A normalização em lote pode ajudar a reduzir a dependência das taxas de aprendizado e inicialização dos pesos iniciais, permitindo que as redes neurais sejam treinadas com taxas de aprendizado mais altas, o que pode acelerar o treinamento.
  - ❖ **Melhoria do desempenho:** A normalização em lote pode ajudar a mitigar o problema da "covariate shift", que é a mudança na distribuição dos dados de entrada durante o treinamento. Isso pode melhorar a estabilidade e a generalização do modelo.
  - ❖ **Regularização implícita:** A normalização em lote também pode atuar como uma forma de regularização implícita, ajudando a reduzir o overfitting ao estabilizar as ativações intermediárias durante o treinamento.



Fonte: <https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338>

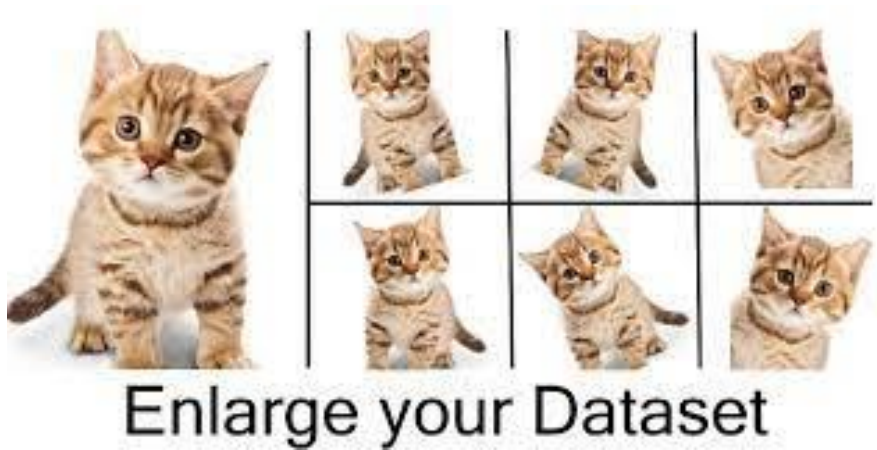
# 4º Callbacks

- ❖ Funcionalidades uteis que podem ser utilizadas durante o processo de treino, como por exemplo:
  - ❖ Parar o treino quando atingir uma determinada acurácia;
  - ❖ Salvar o melhor modelo determinado pelo treino;



The screenshot displays the TensorFlow API documentation for the `tf.keras.callbacks` module. On the left, a sidebar lists various callback classes, including `BackupAndRestore`, `BaseLogger`, `CSVLogger`, `Callback`, `CallbackList`, `EarlyStopping`, `History`, `LambdaCallback`, `LearningRateScheduler`, `ModelCheckpoint`, `ProgbarLogger`, `ReduceLROnPlateau`, `RemoteMonitor`, `SidecarEvaluatorModelExport`, `TensorBoard`, `TerminateOnNaN`, and `experimental`. The main content area shows the module name `Module: tf.keras.callbacks` with buttons for `See Stable` and `See Nightly`. Below this, a description states: "Callbacks: utilities called at certain points during model training." The `Modules` section lists the `experimental` module as the public API for the `tf.keras.callbacks.experimental` namespace. The `Classes` section lists three classes: `class BackupAndRestore` (Callback to back up and restore the training state), `class BaseLogger` (Callback that accumulates epoch averages of metrics), and `class CSVLogger` (Callback that streams epoch results to a CSV file).

# 5° Data Augmentation



Crop



Symmetry



Rotation



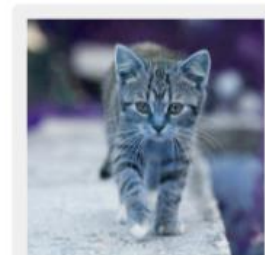
Scale



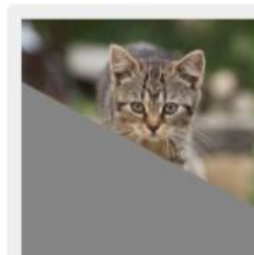
Original



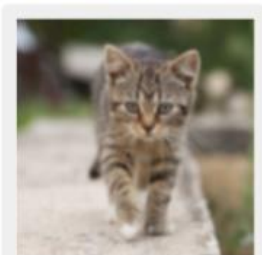
Noise



Hue



Obstruction



Blur



## ❖ Arquitetura de Redes Neurais Convolucionais:

- Existem diversas arquitetura de CNN, cada uma com suas próprias características, principalmente para visão computacional. Mas todas terão em comum camadas de convolução e maxpooling, dropout

## ❖ Por que usar uma CNN já pronta?

- Utilizar uma arquitetura de CNN possibilita reduzir o tempo de pesquisa com o desenvolvimento de novas arquiteturas uma vez que essas arquiteturas já foram sistematicamente revisadas.
- O treinamento de uma boa CNN não é simples, além de muitos dados (milhares de imagens) e muito tempo de processamento.
- Uma forma de contornar esse problema é a utilização de redes pré-treinadas com conjunto de dados de milhares de imagens, o que garante uma boa acurácia.
- É possível ajustar os pesos das últimas camadas da rede para detectar apenas os recursos relevantes para o problema específico.



# 6° Transfer Learning – Exemplos de Arquiteturas

## LeNet

- foi proposta em 1998 e já continha camadas de convolução com filtros 5x5 e passo 1, e agrupamentos com filtros 2x2 com passo 2, intercaladas, seguidas de camadas FC. A sequência de camadas eram: CONV-POOL-CONV-POOL-FC-FC.

## AlexNET

- proposta em 2012, bem mais complexa que a LeNET. Essa arquitetura continha cinco camadas de convolução, batch de tamanho 128, e primeiro uso da função de ativação ReLU.

## VGG

- em 2014 surge a arquitetura VGG com a ideia de filtros menores (3x3) em redes mais profundas com mínimo de 12 convoluções e maxpooling com filtros 2x2. Filtros menores geram menos parâmetros. Porém as camadas FC geravam uma quantidade muito grande de parâmetros e, as convoluções iniciais utilizavam muita memória RAM, tornando essa rede muito pesada.

## GoogleNET

- também em 2014 surgiu a ideia de se utilizar filtros de forma paralela, mais especificamente, fazer uso de uma nova camada chamada Inception, que se tornou elemento básico desta rede, onde tinha-se em sequência, nove módulos do tipo Inception. Este módulo possui convoluções 3x3 e 5x5 precedidas de convoluções 1x1 a fim de se reduzir o custo computacional.

## ResNET

- ou rede residual, proposta em 2015, de forma simplificada, a ideia é de se realizar um curto-circuito a cada duas convoluções, acrescentando um resultado anterior ao resultado futuro. Assim, diferentemente de uma rede tradicional, quanto mais camadas, menor o erro. Porém para os atuais projetos ResNET de 50, 101 e 152 camadas, ao invés de se trabalhar com 2 camadas convolucionais de 3x3, uma é retirada e são inseridas duas convoluções 1x1, diminuindo o custo computacional, como visto na GoogleNET.

Copyright © 2023 Prof. Airton Y. C. Toyofuku

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).