# Projeto de Deep Learning

Bruno Fernandes

UNIVERSIDADE DE PERNAMBUCO

# Conteúdo

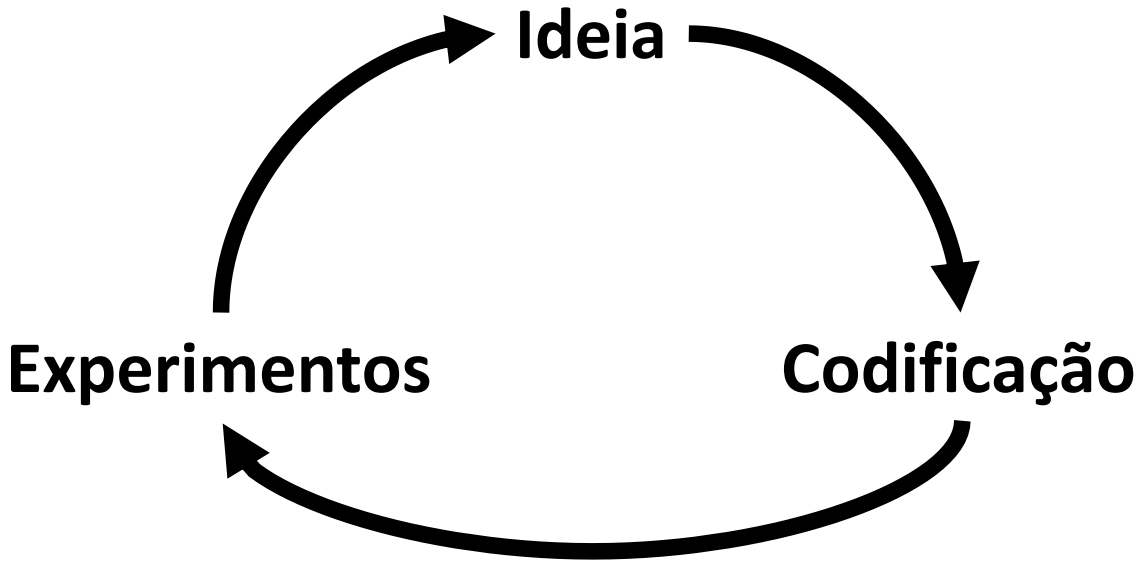
- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Conteúdo

- **Ajuste de hiperparâmetros**
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Ajuste de Hiperparâmetros

- Processo iterativo



# Ajuste de Hiperparâmetros

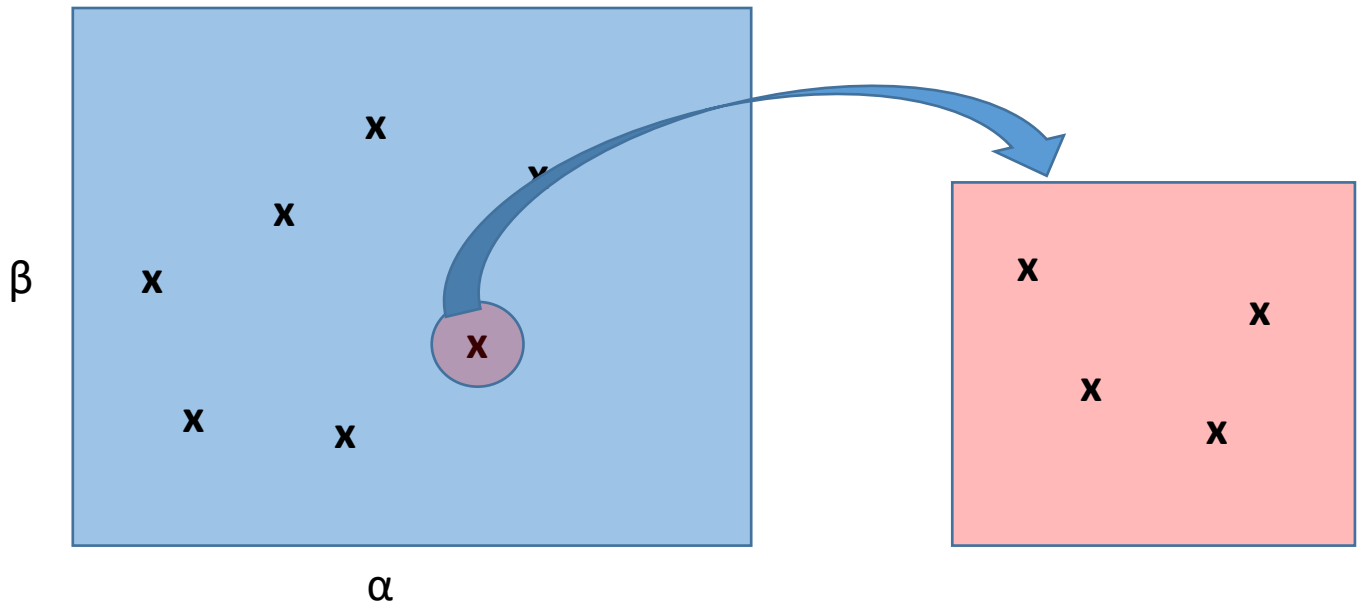
- Prioridades
  1. Taxa de aprendizagem
  2. Termo de momento (β=0,9), tamanho do mini-batch e número de unidades escondidas
  3. Número de camadas e decaimento da taxa de aprendizagem

  Os parâmetros do Adam normalmente não se ajustam ($\beta_1$=0,9, $\beta_2$=0,999, $\beta 3=10^{-8}$)

# Ajuste de Hiperparâmetros

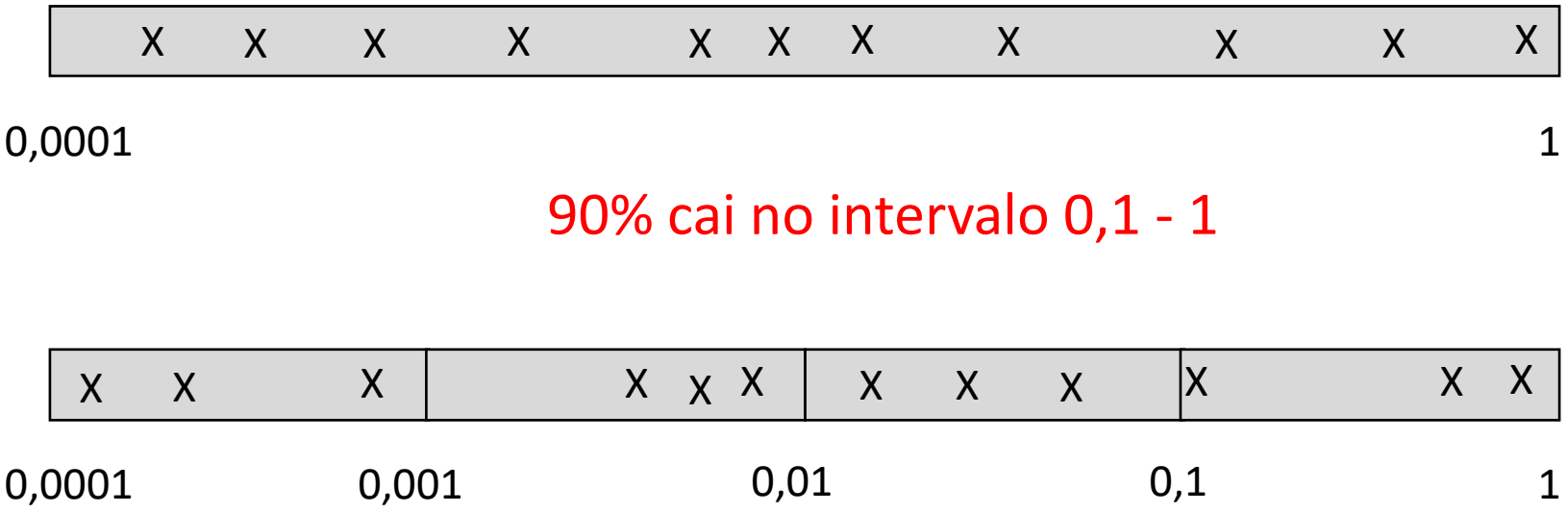
- Selecione valores aleatórios dentro de um espaço de amostragem

- Dê um zoom onde os melhores valores são encontrados e repita o processo



# Ajuste de Hiperparâmetros

- Escolha uma escala apropriada



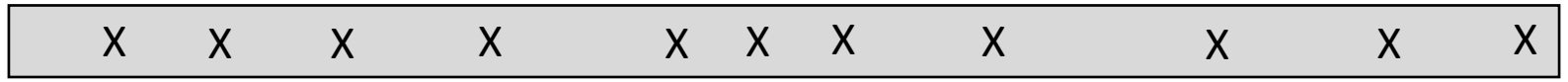

**Em Python**

$r=10^{-4}*np.random.rand()$

$A=10^{r}$

# Ajustando termo do momento

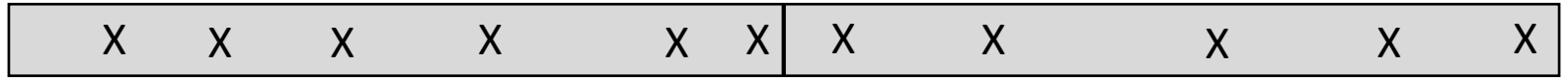
- β=0,9 ... 0,999



0,9 0,999

Errado

- 1-β =0,1 ... 0,001



0,1 $10^{-1}$

$r \epsilon [-3, -1]$

$1-\beta=10^{-r}$

$\beta =1-10^{r}$

0,001 $10^{-3}$

# Abordagens para tunning de parâmetros

# Conteúdo

- Ajuste de hiperparâmetros
- **Ortogonalização**
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Ortogonalização

- Processo para definir o que será ajustado primeiro

Imagina o controle de um carro a partir de uma única função que retorna a velocidade e o ângulo do volante

# Ortogonalização

- Em aprendizagem de máquina, existem quatro objetivos a serem atingidos
  - Ajustar bem no treino
    - Senão, aumente o tamanho da rede ou mude o algoritmo de otimização
  - Ajustar bem no dev
    - Use regularização ou aumente o conjunto de treino
  - Ajustar bem no teste
    - Aumente o conjunto de dev
  - Funcionar bem no mundo real
    - Mude o conjunto de dev ou a função de custo

# Métrica de avaliação com único número

- Exemplo: ao invés de *precision* (p) e *recall* (r), use F1-score$= \frac{2}{\frac{1}{p}+\frac{1}{r}}$
- Outro exemplo: média de valores

- A tarefa nem sempre é simples: **como combinar acurácia e tempo de execução?**

# Combinar múltiplos fatores

- Estabeleça uma das variáveis para ser maximizada e inclua restrições nas demais
- No caso da acurácia e do tempo
  - Maximizar acurácia <- otimizar
  - Sujeito a restrição de tempo menor que *T*ms <- satisfazer
- Com N métricas
  - 1 otimizar
  - N-1 satisfazer

# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- **Bias, variância e performance humana**
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Bias e Variância

- Se o erro no treino é alto, diz-se que o problema tem um bias alto (**underfitting**)
- Se o erro no dev é alto, diz-se que o problema tem uma variância alta (**overfitting**)
- **Como definir se o erro do treino ou do dev é pior?**

UNIVERSIDADE DE PERNAMBUCO

# Performance Humana

- Serve como base para estimar o erro de bayes

- Enquanto o modelo for pior que o humano, faça:
  - Pegue mais dados rotulados de humanos
  - Pegue insights dos humanos sobre os erros (análise qualitativa manual)
  - Analise a relação dos erros nos conjuntos de treino e dev (bias e variância)

# Bias evitável

| | Problema-1 | Problema-2 |
|---|---|---|
| Humano | 1% | 7,5% |
| Erro no treino | 8% | 8% |
| Erro no dev | 10% | 10% |

bias

variância

- Problema-1
  - Foco no bias
- Problema-2
  - Foco na variância

# Bias ou Variância

- Bias
  - Treinar uma rede maior
  - Treinar por mais tempo/melhor: momento, rmsprop, adam
  - Pesquisa por hiperparâmetros/arquitetura da rede
- Variância
  - Mais dados
  - Regularização: L2, dropout, data augmentation
  - Pesquisa por hiperparâmetros/arquitetura da rede

# Análise do Erro

- Verificar qual tipo do erro é mais comum para concentrar os esforços
- Problema de reconhecimento de gato
- Erros

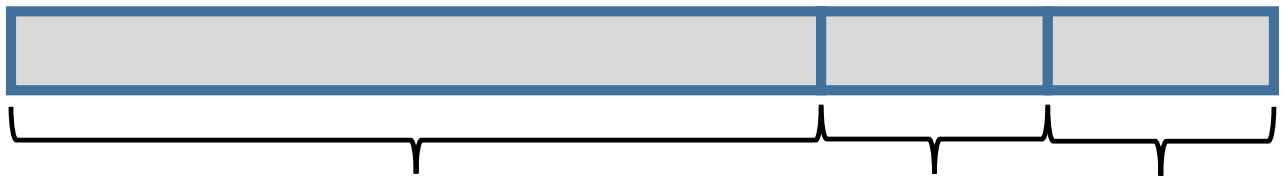
| Imagem | Cachorro | Grandes felinos | Borrada | |
|---|---|---|---|---|
| 1 | X | | | Pitbull |
| 2 | | | X | |
| 3 | | X | X | Dia chuvoso no zoológico |
| ... | ... | ... | ... | |
| % do total | 8% | 43% | 61% | |

# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- **Distribuição treino/teste**
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Distribuição treino/teste

- Anteriormente




ou 70%/30%

# Distribuição treino/teste

- Big data (milhões de dados)

98%
**Treino**

1% 1%
**DevTeste**

ou 98,5%/0,4%/0,1%

**É importante que os conjuntos de dev e teste venham da mesma distribuição!**

# Dados de distribuições diferentes

- Suponha o problema para reconhecer gatos com a câmera do celular
- Entretanto, para treinar seu modelo de maneira mais eficaz, você coletou também imagens da Web
- Você tem 200 mil imagens coletadas da Web e 10 mil imagens coletadas pelo app mobile
- Como deve ser separado os conjuntos de treino, dev e teste?

# Dados de distribuições diferentes

- Opção 1: Junta tudo em única conjunto, embaralha e separa
  - Ruim porque o alvo será majoritariamente imagens da Web

- Opção 2: separa 5 mil imagens aleatórias do app para treino e as outras 5 mil para dev e teste, as 200 mil da Web ficam para treino
  - Existe o risco das imagens no conjunto de dev/teste representarem um problema bem mais difícil e, consequentemente, obter uma taxa bem pior, mas que não quer dizer o problema da variância
  - Solução: crie um novo conjunto treino-dev com a mesma distribuição do treino para avaliar a convergência do modelo

# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- **Transfer learning**
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Transfer Learning

- Substitui camadas de um modelo já treinado para um problema por novas camadas a serem treinadas para um novo problema
- Pode treinar só as novas camadas ou a rede toda (*pre-training + fine tunning*)
- Pode reinicializar os pesos das últimas camadas ou mesmo substitui-las
- Sempre deve substituir a camada de softmax
- Faz sentido quando se tem muitos dados no problema de origem e bem menos no de destino
- Via de regra, sempre opte por transfer learning
  - A não ser que o conjunto de dados seja absurdamente grande, assim como os recursos computacionais

# Conteúdo

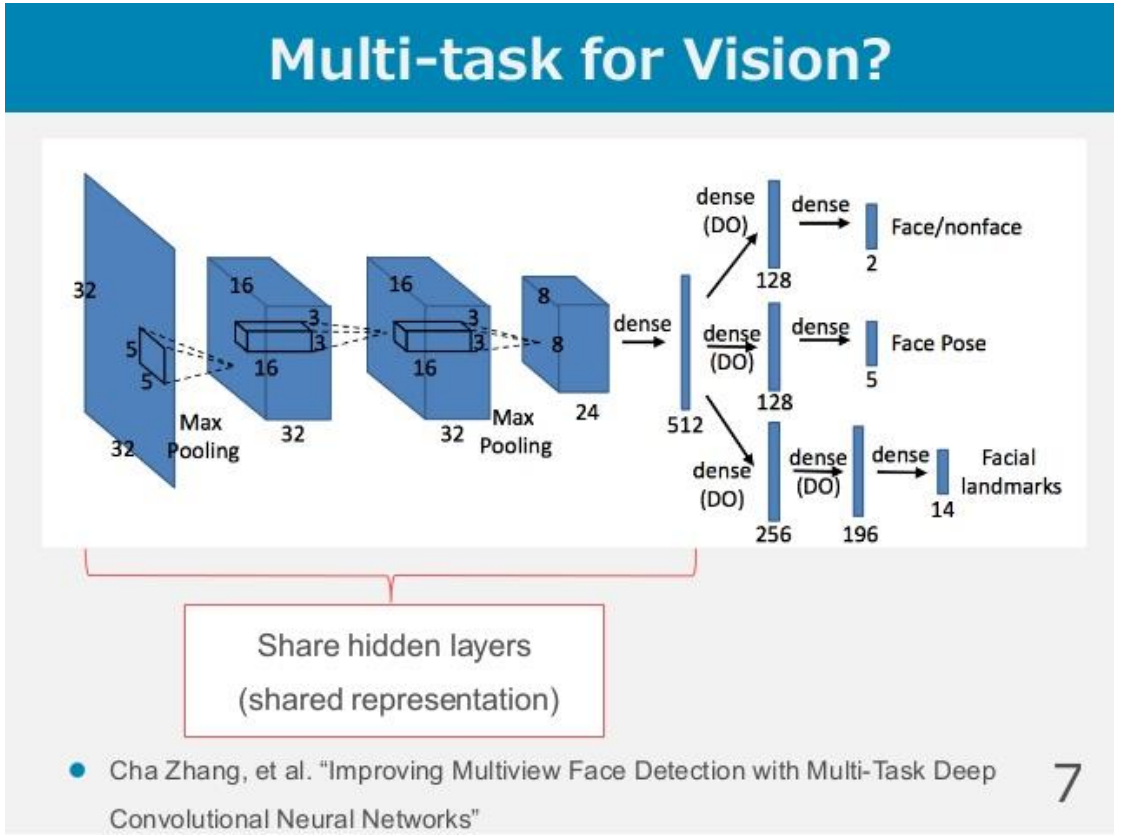
- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- **Multi-task learning**
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Multi-task learning

- Pode ser mais fácil treinar uma rede para várias tarefas juntas do que uma rede por tarefa
- Dessa forma, pode indicar que numa mesma imagem tem um gato e um cachorro, mas nenhum coelho
- Não se usa softmax na saída
- No caso de algumas saídas não informar se tem o label ou não, simplesmente  ignore-as no cálculo do erro

# Multi-task learning

- Faz sentido quando
  - As tarefas compartilham características de baixo nível
  - Os conjuntos de dados das tarefas são similares
  - Pode treinar uma rede neural para rodar bem em todos os problemas (normalmente, só não dá certo se a rede não for grande o suficiente)



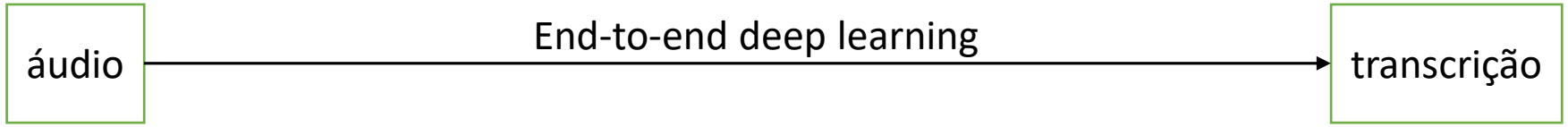
# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- **End-to-end learning**
- Data augmentation
- Prática: Transfer Learning
- O Estado da Visão Computacional

# End-to-end Deep Learning

- Abordagem tradicional envolve uma série de passos até o resultado final


- Uma forma de simplificar o sistema é deixar que a rede aprenda todo o processo




  - Só funciona bem se tiver muitos dados, com poucos é melhor a abordagem tradicional

# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- **Data augmentation**
- Prática: Transfer Learning
- O Estado da Visão Computacional

# Data augmentation

- Estratégia para aumentar a quantidade de padrões de treino apresentados a rede
  - **Espelhamento**
  - **Recorte aleatório**
  - Rotação
  - *Shearing*
  - *Warping*
  - *Color shift*
  - PCA
- Na prática: https://keras.io/preprocessing/image/

# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- **Prática: Transfer Learning**
- O Estado da Visão Computacional

# Prática: Transfer Learning

- Criação do modelo

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu', data_format='channels_first',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

# Prática: Transfer Learning

- Treino e teste

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

# Prática: Transfer Learning

- Congelando pesos e trocando a última camada

```
for layer in model.layers:
    layer.trainable = False
modelTL = model
modelTL.pop()
modelTL.add(Dense(2, activation='softmax'))
```

# Prática: Transfer Learning

- Treinando e testando

```
modelTL.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
batch_size = 32
epochs = 120
modelTL.fit(cat_train_x, cat_train_y,
          batch_size=32,
          epochs=epochs,
          verbose=1,
          validation_data=(cat_test_x, cat_test_y))
score = modelTL.evaluate(cat_test_x, cat_test_y, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

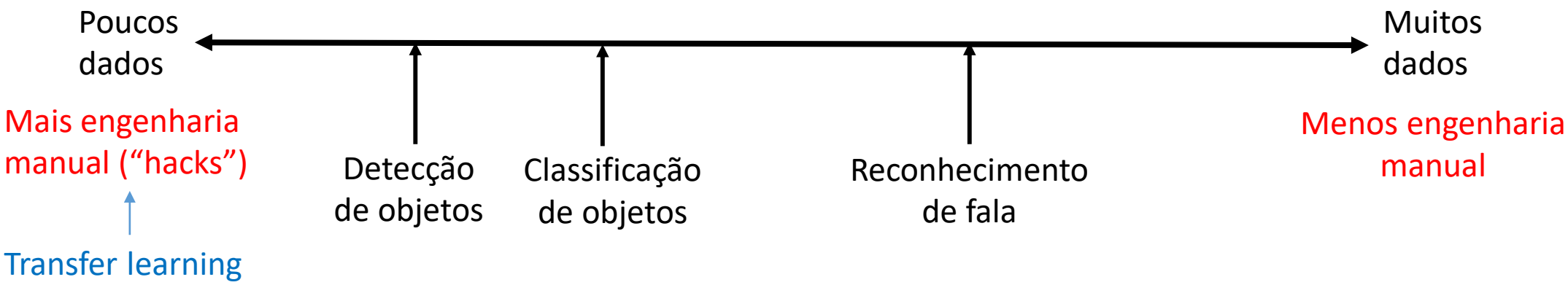
# Prática: Transfer Learning

- Existem também outras formas...
- Vamos olhar o código

# Conteúdo

- Ajuste de hiperparâmetros
- Ortogonalização
- Bias, variância e performance humana
- Distribuição treino/teste
- Transfer learning
- Multi-task learning
- End-to-end learning
- Data augmentation
- Prática: Transfer Learning
- **O Estado da Visão Computacional**

# O Estado da Visão Computacional




- Fontes de conhecimento:
  - Dados rotulados
  - Características construídas a mão/arquitetura da rede/outros componentes

# Dicas para benchmarks

- Ensembling
  - Comitê com médias das saídas (3 a 15 redes)

- Multi-crop em tempo de teste
  - Espelha-se a imagem de teste e pega 10 crops (ou menos) da imagem e tira a média dos resultados




# Projeto de Deep Learning

Bruno Fernandes

UNIVERSIDADE DE PERNAMBUCO