# Deep Learning através das abstrações

Revisão Keras e Deep Learning

# Models: Sequential

- Simples de usar
- Orientado a objetos
- Limita-se a redes que são sequenciais/caminhos

# Models: Functional

- Mais flexibilidade nos modelos, podem ser DAGs
- Estilo mais funcional
- Podemos facilmente pegar a saída de camadas intermediárias
- Requer um pouco mais de adaptação
    - Tipo usar o Input

# Layers: Dense

- Camadas densas, todos os neurônios se ligam a todos os outros
- Modelo mais simples
- Número elevado de ligações ((input+1)*output)
- Parâmetros principais:
  - units : número de neurônios na saída
  - activation: Função de ativação
  - Input_shape: Dimensão da entrada, só é necessário na primeira

# Layers: Conv2D

- Camadas convolucionais, aprendem filtros
- Bidimensionais
- Bons para imagens
- Parâmetros principais:
  - filters : número de filtros aprendidos
  - kernel_size: Tamanho dos filtros aprendidos
  - activation: Função de ativação
  - strides: tamanho do stride em cada direcão
  - padding: same or valid
  - Input_shape: 4D tensor with shape: `(batch, rows, cols, channels)`

# Layers: MaxPooling2D

- Reduz dimensão e detalhes da imagem
- Bidimensional
- Parâmetros principais:
  - pool_size: fator de down_scaling
  - strides: quanto o filtro anda em cada direção
  - padding: same ou valid

# Layers: Dropout

- Aplica dropout
- Durante treino anula a ativação de p neurônios na camada antes dela
- Durante o teste multiplica todo mundo por 1 - p
- Parâmetros principais:
  - rate: o p
  - seed: a semente do gerador aleatório

# Layers: Batch Normalization

- Aplica batch normalization nas ativações da camada anterior
- Muitos parâmetros complexos, acho que até a inception quase usa o padrão
  - scale: diz se vamos multiplicar pelo desvio padrão aprendido pela camada
  - axis: explicita o eixo que deve ser normalizado

# Layers: Activation e Flatten

- Activation aplica uma função de ativação nos neurônios antes dele
- Flatten transforma uma entrada multidimensional em um vetor

# Losses: Regressão

- Mean Squared Error
- Mean Absolute Error
- Mean Absolute Percentage Error
- Mean Squared Logarithmic Error
- Logcosh
    - Mean squared error menos afetado por outliers

# Losses: Classificação

- Categorical Cross Entropy
- Sparse Categorical Cross Entropy
- Kullback-leibler divergence
- Hinge
- Squared Hinge
- Categorical Hinge
- Cosine proximity

# Deep Learning no mundo real

- Temos mais dados
  - Digitalização de tudo
- Mais poder computacional
  - GPUs estão bem mais acessíveis
  - Possível alugar

# Deep Learning no mundo real

- Não é uma silver bullet/martelo de ouro
  - Mas vale a pena testar se tiver os recursos
- Idealmente ter um dataset grande
- Sempre bom ter um baseline antes de aplicar

# Deep Learning no mundo real

- Coleta de dados
  - Depende da complexidade do seus problema
    - Quantas features cada exemplo tem?
    - Quantas classes você quer prever?
    - Quão separável são essas classes?
  - Sempre bom comparar com datasets conhecidos
- Resolução das imagens
  - Procure a menor possível
    - Dado que ainda é possível diferenciar as classes
    - Se tiver muito poder computacional pode deixar maior mesmo
- Divida os conjuntos (treino, teste e validação)

# Deep Learning no mundo real

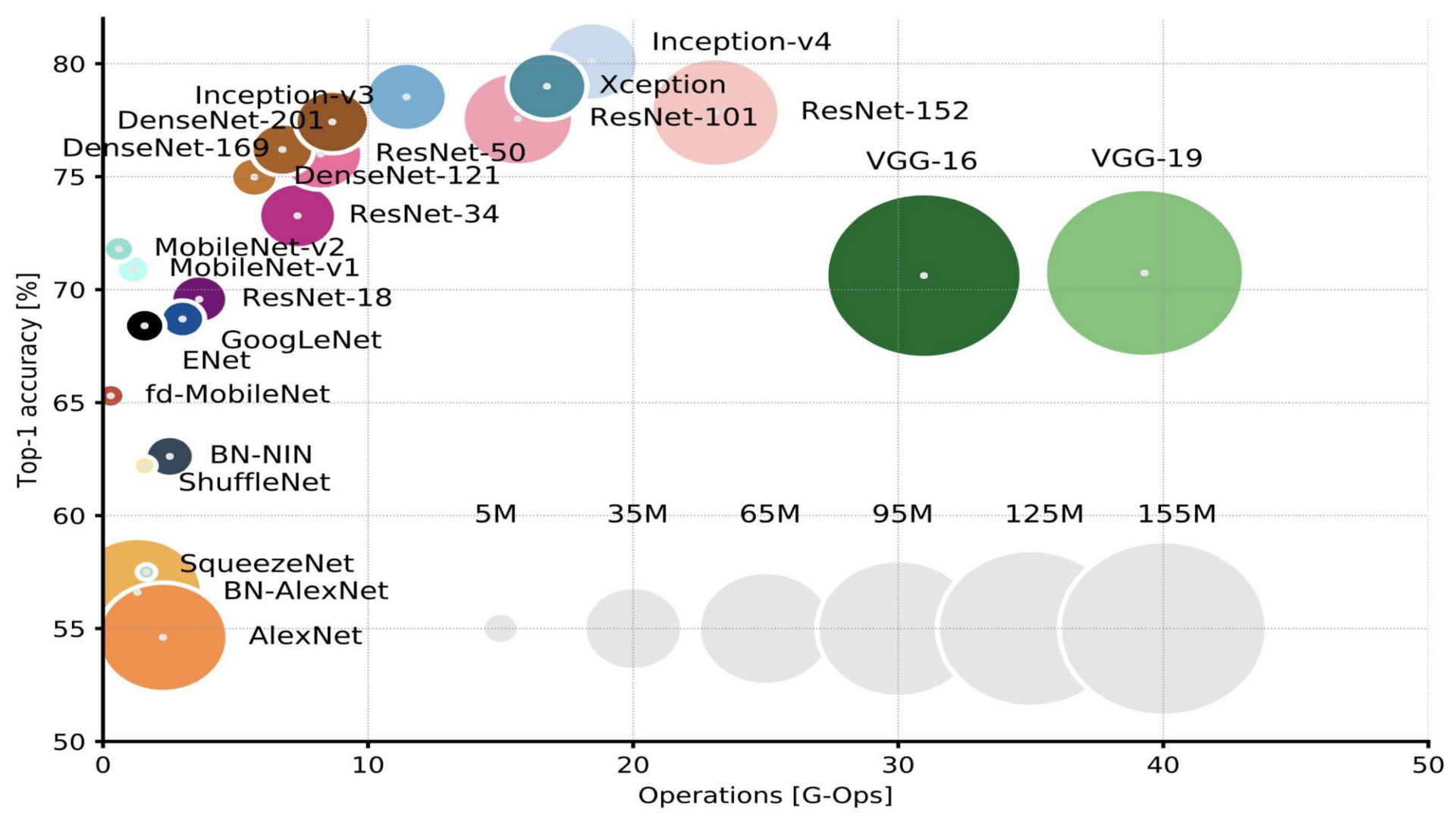
- Como lidar com datasets pequenos?
  - Data augmentation
  - Transfer Learning
    - Pre trained models
- Em alguns casos pode ser interessante fazer o seu próprio modelo
  - Otimizar hiperparâmetros
  - Validação cruzada

# Image Augmentation

- Útil para aumentar o seu dataset
- Maior variedade
- Maior generalização
  - Um possível efeito é treinar uma rede invariante as transformações dadas
- Também usado para lidar com classes desbalanceadas
- Necessário pensar quais transformações serão aplicadas

# Transfer Learning

- Extremamente usado em aplicações reais
- Analisar o número de parâmetros e acurácia de cada rede
- Em geral dois modos de fazer
  - Treinar apenas um classificador linear na última camada
    - Usado principalmente para datasets parecidos
  - Re treinar (fine tune) a rede para o seu dataset
    - Pode ser mais preciso
    - Porém com poucos dados pode gerar overfitting


Inception-v4
Xception
Inception-v3
ResNet-101
ResNet-152
DenseNet-201
DenseNet-169
ResNet-50
VGG-16
VGG-19
DenseNet-121
ResNet-34
MobileNet-v2
MobileNet-v1
ResNet-18
GoogLeNet
ENet
fd-MobileNet
BN-NIN
ShuffleNet
5M
35M
65M
95M
125M
155M
SqueezeNet
BN-AlexNet
AlexNet
Top-1 accuracy [%]
80
75
70
65
60
55
50
0
10
20
30
40
50
Operations [G-Ops]

# Resources

- Documentação do Keras
- Post sobre image augmentation
- Blog do Keras
- Dicas de Transfer Learning
- Alguns cuidados necessários