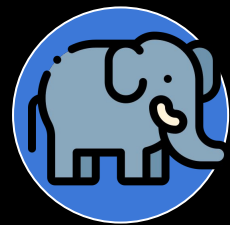


“Fica, vai ter bolo”.

Inicialização de Pesos e Transfer Learning

Paulo Ricardo Lisboa de Almeida

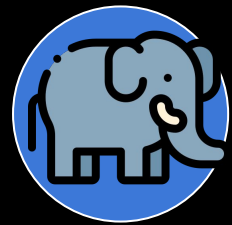




O Elefante na Sala

Considere a seguinte CNN:

```
class custom_3_layer(nn.Module):  
    def __init__(self):  
        super(custom_3_layer, self).__init__()  
  
        self.conv1 = nn.Conv2d(in_channels=INPUT_SHAPE[0], out_channels=32, kernel_size=3, stride=1, padding=1)  
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)  
  
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=0)  
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)  
  
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=0)  
        ...
```

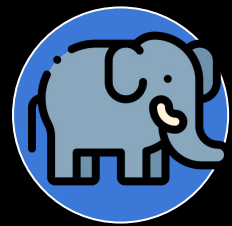


O Elefante na Sala

Considere a seguinte CNN:

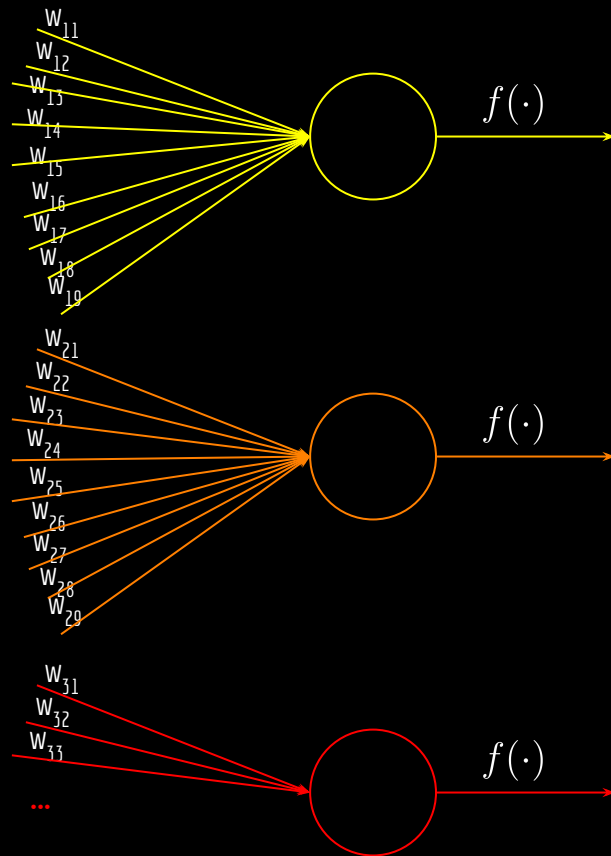
```
class custom_3_layer(nn.Module):  
    def __init__(self):  
        super(custom_3_layer, self).__init__()  
  
        self.conv1 = nn.Conv2d(in_channels=INPUT_SHAPE[0], out_channels=32, kernel_size=3, stride=1, padding=1)  
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)  
  
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=0)  
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)  
  
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=0)  
        ...
```

Quantos filtros de convolução existem na primeira camada, e qual o tamanho dos filtros?



O Elefante na Sala

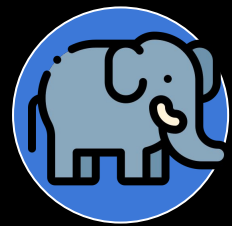
São 32 filtros de 3×3 , com 9 parâmetros cada (mais o bias).



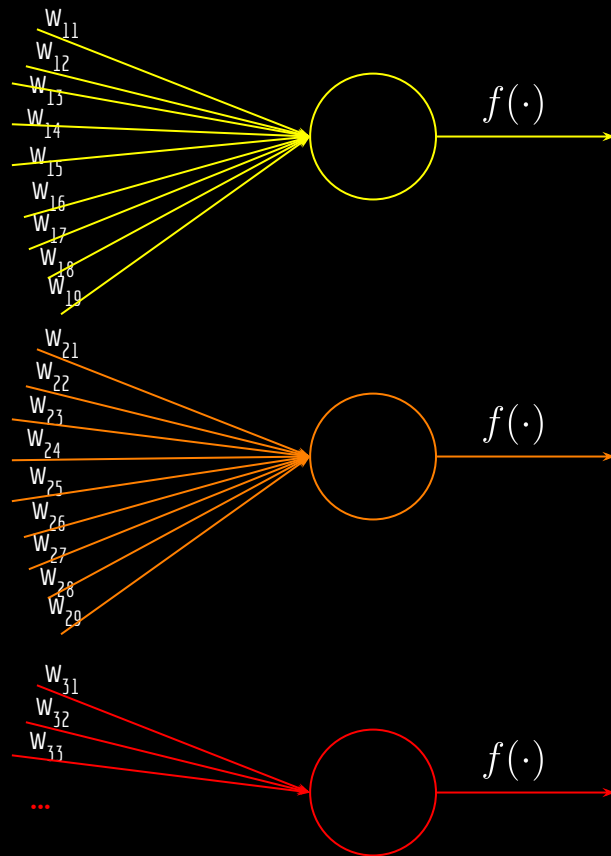
W_{11}	W_{12}	W_{13}
W_{14}	W_{15}	W_{16}
W_{17}	W_{18}	W_{19}

W_{21}	W_{22}	W_{23}
W_{24}	W_{25}	W_{26}
W_{27}	W_{28}	W_{29}

W_{31}	W_{32}	W_{33}
W_{34}	W_{35}	W_{36}
W_{37}	W_{38}	W_{39}



O Elefante na Sala



Durante o treinamento, era para todos os filtros convergirem para os mesmos pesos, não? Como filtros diferentes aprendem coisas diferentes?

W_{11}	W_{12}	W_{13}
W_{14}	W_{15}	W_{16}
W_{17}	W_{18}	W_{19}

= ???

W_{21}	W_{22}	W_{23}
W_{24}	W_{25}	W_{26}
W_{27}	W_{28}	W_{29}

= ???

W_{31}	W_{32}	W_{33}
W_{34}	W_{35}	W_{36}
W_{37}	W_{38}	W_{39}

Faça você mesmo

Execute o exemplo disponibilizado no Google Colab.

É a mesma rede de aulas passadas, mas agora todos os filtros de convolução 3×3 são inicializados com os mesmos valores.

Todos biases inicializados com 1.

Faça você mesmo

É muito provável que todos os kernels de convolução tenham convergido para os mesmos valores.

Ter múltiplos kernels se tornou inútil.

```
... [[[-0.0698,  0.0933, -0.3015],  
       [-0.1609, -0.0964, -0.3220],  
       [-0.2577, -0.3699, -0.2675]]],  
  
     [[[-0.0698,  0.0933, -0.3015],  
       [-0.1609, -0.0964, -0.3220],  
       [-0.2577, -0.3699, -0.2675]]],  
  
     [[[-0.0698,  0.0933, -0.3015],  
       [-0.1609, -0.0964, -0.3220],  
       [-0.2577, -0.3699, -0.2675]]] ...
```

Quebra de simetria

Devemos considerar a quebra de simetria durante a inicialização dos pesos.

Se todos os pesos possuem os mesmos valores inicialmente, a rede é simétrica, eles contribuirão igualmente para o erro, e serão atualizados com os mesmos valores.

Quebra de simetria

É necessário inicializar os pesos aleatoriamente.

Mas qualquer aleatório também não serve.

Dependendo da complexidade da rede, ou da disponibilidade de recursos, podemos ter múltiplas inicializações, e escolher a que converge melhor.

Inicialização

Geralmente os pesos são inicializados através de:

Distribuição Normal no intervalo $[-\epsilon, \epsilon]$.

Gaussiana de média zero $\mathcal{N}(0, \epsilon^2)$.

A escolha de ϵ varia de acordo com o método.

Exemplo - Inicialização de He

Em uma rede que utiliza a função de ativação ReLU e similares desejamos que a variância dos dados que entram nos neurônios:

- Não caiam para zero.
- Não subam muito.

He, K., Zhang, X., Ren, S., & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE international conference on computer vision*, 2015.



This CVF paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the version available on IEEE Xplore.

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research

Abstract

Rectified activation units (rectifiers) are essential for state-of-the-art neural networks. In this work, we study rectifier neural networks for image classification from two aspects. First, we propose a Parameterized Rectified Linear Unit (PReLU) that generalizes the traditional rectified unit. PReLU supports model training with nearly zero extra computational cost and little overhead risk. Second, we derive a robust initialization method that particularly considers the rectifier nonlinearities. This method enables us to train extremely deep rectified models directly from scratch and to initialize deeper or wider network architectures. Based on the Synapse Activation and Advanced Initialization, we achieve 4.0% top-5 error on the ImageNet 2012 classification dataset. This is 20% relative improvement over the ILSVRC 2014 winner (GoogLeNet, 4.9% [14]). To our knowledge, our result is the first to surpass the reported human-level performance (5.1% [26]) on this dataset.

1. Introduction

Conventional neural networks (CNNs) [19, 18] have demonstrated recognition accuracy better than or comparable to humans in several visual recognition tasks, including recognizing traffic signs [15], faces [14, 32], and hand-written digits [3, 36]. In this work, we present a result that surpasses the human-level performance reported by [26] on a more generic and challenging recognition task, the classification task in the 1000-class ImageNet dataset [24].

In the last few years, we have witnessed tremendous improvement in recognition performance, mainly due to advances in two technical directions: building more powerful models and designing effective strategies against overfitting. On one hand, neural networks are becoming more capable of fitting training data. Because of increased complexity (e.g., increased depth [29, 27], enlarged width [17, 28], and the use of deeper models [17, 28, 2, 20]), new nonlinear activation units [24, 23, 38, 22, 31, 30], and sophisticated layer designs [13, 12]. On the other hand, better generalization is achieved by effective regularization

techniques [13, 30, 38, 36], aggressive data augmentation [10, 14, 29, 31], and large-scale data [4, 26].

Among these advances, the rectifier neurons [24, 9, 23, 38], e.g., Rectified Linear Unit (ReLU), is one of several keys to the recent success of deep networks [18]. It expedites convergence of the training procedure [14] and leads to better robustness [12, 9, 23, 30] than conventional sigmoid-like units. Despite the prevalence of rectifier networks, recent improvements of models [17, 28, 12, 29, 31] and theoretical guidelines for training them [8, 27] have rarely focused on the properties of the rectifiers.

Unlike traditional sigmoid-like units, ReLU is not a symmetric function. As a consequence, the mean response of ReLU is always no smaller than zero; besides, even assuming the input/bias/weights are subject to symmetric distributions, the distributions of responses can still be asymmetric because of the behavior of ReLU. These properties of ReLU influence the theoretical analysis of convergence and empirical performance, as we will demonstrate.

In this paper, we investigate neural networks from two aspects particularly driven by the rectifier properties. First, we propose a new extension of ReLU, which we call Parameterized Rectified Linear Unit (PReLU). This activation function properly learns the parameters of the rectifiers, and improves accuracy at negligible extra computational cost. Second, we study the difficulty of training rectified models that are very deep. By explicitly modeling the non-linearity of nonlinear ReLU/PReLU, we derive a theoretically sound initialization method, which helps with convergence of very deep models (e.g., with 30 middle layers) trained directly from scratch. This gives us more flexibility to explore more powerful network architectures.

On the 1000-class ImageNet 2012 dataset, our network leads to a single-model result of 3.7% top-5 error, which surpasses all single-model results in ILSVRC 2014. Further, our multi-model result achieves 4.0% top-5 error on the test set, which is a 20% relative improvement over the ILSVRC 2014 winner (GoogLeNet, 4.9% [14]). To the best of our knowledge, our result surpasses for the first time the reported human-level performance (5.1% in [26]) of a dedicated individual human on this recognition challenge.

revised in Feb. 2015.

Exemplo - Inicialização de He

Considerando que temos M conexões com a camada anterior da camada atual, o valor ϵ é dado por:

$$\epsilon = \sqrt{\frac{2}{M}}$$

Outro exemplo

Inicialização de Xavier.

$$\epsilon = \sqrt{\frac{2}{M+N}}$$

Onde N é o número de conexões de saída.

Glorot, X., & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Understanding the difficulty of training deep feedforward neural networks

Xavier Glorot

UMR5142, Université du Montréal, Montréal, Québec, Canada

Yoshua Bengio

Abstract

Whether before 2006 it appears that deep multi-layer neural networks were not successfully trained, since then several algorithms have been shown to successfully train them, with experimental results showing the superiority of deeper to less deep architectures. All these experimental results were obtained with new initialization or training mechanisms. Our objective here is to understand better why standard gradient descent from random initialization is doing so poorly with deep neural networks, to better understand those recent initiatives, successes and help design better algorithms in the future. We first observe the influence of the non-linear activation function. We find that the logistic sigmoid activation is ill-suited for deep networks with random initialization because of its mean value, which can drive especially the top hidden layer into saturation. Surprisingly, we find that saturated units can move out of saturation by themselves, albeit slowly, and triggering the plateau recovery seen when training neural networks. We find that a new bio-inspired initialization scheme can often be beneficial. Finally, we study how activations and gradients can saturate layers during training, with the idea that training may be more difficult when the singular values of the Jacobian associated with each layer are far from 1. Based on these considerations, we propose a new initialization scheme that brings substantially faster convergence.

1 Deep Neural Networks

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. They include convolutional layers (LeCun et al., 1998), restricted Boltzmann machines (Salakhutdinov and Hinton, 2006), deep belief networks (Bengio et al., 2006), and deep restricted Boltzmann machines (Bengio et al., 2007).

learning methods for a wide array of deep architectures, including neural networks with many hidden layers (Vahdat et al., 2006) and graphical models with many levels of hidden variables (Ghahramani et al., 2006), among others (Glorot et al., 2008; Bengio et al., 2009). Much research has recently been devoted to these (see Bengio, 2009 for a review). Success of these methods of deep learning originates from biology and human cognition, and because of empirical success in various domains (Bengio et al., 2007; LeCun et al., 2007; Vincent et al., 2006) and natural language processing (Collobert & Weston, 2008; Mnih & Hinton, 2009). Theoretical results reviewed and discussed by Bengio (2009), suggest that in order to learn the kind of complex and non-linear functions that can represent high-level abstractions (e.g. in vision, language, and other AI-level tasks), one may need deep architectures.

Most of the recent experimental results with deep architectures are obtained with models that can be trained using deep supervised neural networks, but with initialization or training schemes different from the classical feedforward neural networks (Rumelhart et al., 1986). Why are these new algorithms working so much better than the standard random initialization and gradient-based optimization of a supervised training, considered? Part of the answer may be found in recent analyses of the effects of supervised pre-training (Bengio et al., 2009), showing that it gives a useful pre-training that regularizes the parameters in a "better" kind of attraction of the optimization procedure, corresponding to an optimal initialization associated with these generalization. But earlier work (Bengio et al., 2007) had shown that overly simple supervised pre-training procedures would give better results. So here instead of focusing on what supervised pre-training or semi-supervised criteria bring to deep architectures, we focus on analyzing what may be going wrong with good old (but deep) multi-layer neural networks.

Our analysis is driven by investigative experiments to measure activations (measured by saturation or hidden units) and gradients, across layers and across training iterations. We also evaluate the effects on these of choices of activation functions (with the idea that a single affine saturation and initialization procedure (like unsupervised pre-training in a particular form of initialization and it has a drastic impact).

Transfer Learning

Ideia: usar algum dataset grande para treinar os filtros de convolução da rede.

O dataset não precisa conter os mesmos dados da tarefa alvo.

Exemplo: treinamos uma rede para reconhecer cães, e depois a ajustamos para reconhecer gatos.

Transfer Learning

Ideia: usar algum dataset grande para treinar os filtros de convolução da rede.

O dataset não precisa conter os mesmos dados da tarefa alvo.

Exemplo: treinamos uma rede para reconhecer cães, e depois a ajustamos para reconhecer gatos.

Assume-se que os filtros de convolução podem ser usados para outras tarefas.

Retreinamos a camada de classificação da rede, que usa como entrada os filtros de convolução já treinados.

Transfer Learning

Ideia: usar algum dataset grande para treinar os filtros de convolução da rede.

O dataset não precisa conter os mesmos dados da tarefa alvo.

Exemplo: treinamos uma rede para reconhecer cães, e depois a ajustamos para reconhecer gatos.

Assume-se que os filtros de convolução podem ser usados para outras tarefas.

Retreinamos a camada de classificação da rede, que usa como entrada os filtros de convolução já treinados.

Pode ser visto como uma técnica de **regularização**.

Estamos usando dados de outras tarefas para treinar um modelo para a tarefa alvo.

Exemplo

No exemplo, vamos usar a MobileNetV3 small.

Veja como a rede funciona nos artigos.

Especial atenção para as camadas de *Bottleneck*.

Misturam convoluções com expansões sem uso de não linearidades para dimensões maiores.

Sandler, M., et al. Mobilenetv2: Inverted residuals and linear bottlenecks. *IEEE conference on computer vision and pattern recognition*. 2018.

Howard, Andrew, et al. Searching for mobilenetv3. *IEEE/CVF international conference on computer vision*. 2019.

MobileNetV2: Inverted Residuals and Linear Bottlenecks

Mark Sandler Andrew Howard Mengshu Zhu Andrew Zhmoginov Liang-Chieh Chen
Google Inc.
(sandler, howard, mengshu, azhmoginov, lichen)@google.com

Abstract

In this paper we describe a new mobile architecture, MobileNetV2, that improves the state of the art performance of mobile models on multiple tasks and benchmarks as well as serves as a picture of different model sizes. We also describe efficient ways of applying these mobile models to object detection in a novel framework, we call SSDLite. Additionally, we demonstrate how to build mobile semantic segmentation models through a reduced form of DeepLabv3 which we call Mobile DeepLabv3.

It is based on an inverted residual structure where the residual connections are between the low bandwidth layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. Additionally, we find that it is important to remove non-linearity in the narrow layers in order to maximize representational power. We demonstrate that this improves performance and provide an intuition that led to this design.

Finally, our approach allows decoupling of the performance demands from the expressiveness of the transformation, which provides a convenient framework for further analysis. We measure our performance on ImageNet [1] classification, COCO object detection [2], VOC image segmentation [3]. We evaluate the trade-off between accuracy, and number of operations measured by multi-scale (MAs) as well as actual latency, and the number of parameters.

1. Introduction

Neural networks have revolutionized many areas of research intelligence, enabling superhuman accuracy for challenging image recognition tasks. However, the drive to improve accuracy often comes at a cost: modern state of the art networks require high computational resources beyond the capabilities of many mobile and embedded

applications.

This paper introduces a new neural network architecture that is specifically tailored for mobile and resource constrained environments. The network explores the state of the art for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy.

The main contribution is a novel layer module, the inverted residual with linear bottleneck. This module takes an input as a low dimensional input representation which is first expanded to a high dimensional and filtered with a lightweight depthwise convolution. Features are subsequently projected back to a low dimensional representation with a linear connection. The official implementation is available as part of TensorFlowSlim mobile library [4].

This module can be efficiently implemented using standard operations in any modern framework and allows our models to beat state of the art along multiple performance points using standard benchmarks. Furthermore, the convolutional module is particularly suitable for mobile devices, because it allows to significantly reduce the memory footprint needed during inference by never fully materializing large intermediate tensors. This reduces the need for main memory access in many embedded hardware designs, that provide small amounts of very fast on-chip controlled cache memory.

2. Related Work

Testing deep neural architectures to strike an optimal balance between accuracy and performance has been an area of active research for the last several years. While neural architecture search and improvements in training algorithms, carried out by numerous teams, have led to dramatic improvements over early designs such as AlexNet [5], VGGNet [6], GoogLeNet [7], and ResNet [8]. Recently there has been lots of progress in efficient neural network exploration without needing to train the data to be used to be evaluated. Advances in neural network efficiency not only improve user experience via higher accuracy and lower latency, but also

Searching for MobileNetV3

Andrew Howard¹ Mark Sandler¹ Grace Chen¹ Liang-Chieh Chen¹ Bo Chen¹ Mingxing Tan¹
Yuhang Yang¹ Yihui Chen¹ Renshuo Peng¹ Yipeng Wu¹ Quan X. Li² Hartwig Adam¹
(howard, sandler, chen, howard, mengshu, azhmoginov, yipeng, yuhang, peng, wei, qili, lian)@google.com

Abstract

We present the next generation of MobileNets based on a combination of complementary search techniques as well as a novel architecture design. MobileNetV3 is used as mobile phone CPUs through a combination of hardware-aware neural architecture search (NAS) complemented by the Nelderley algorithm and then subsequently improved through neural architecture evolution. This paper starts the exploration of few automated search algorithms and neural network designs can work together to harness complementary approaches improving the overall state of the art. Through this process we create six new MobileNet models for us: Large, MobileNetV3 Large and MobileNetV3 Small which are targeted for high end low resource use cases. These models are then adapted and applied to the tasks of object detection and semantic segmentation. For the task of semantic segmentation or any dense pixel prediction, we propose a new efficient segmentation decoder, Lite-BiSeNet. When applied to ImageNet classification, we achieve new state of the art results for mobile classification, detection and segmentation. MobileNetV3 Large is 3.2% more accurate on ImageNet classification while reducing latency by 20% compared to MobileNetV2. MobileNetV3 Small is 4.6% more accurate compared to a MobileNetV2 model with comparable latency. MobileNetV3 Large detection is only 25% faster than MobileNetV2 R-50FP at similar accuracy for Cityscapes segmentation.

Figure 1: The trade-off between FLOPs and top-1 accuracy.

Figure 1: The trade-off between FLOPs and top-1 accuracy. All models use the input resolution 224x224. V3 Large and V3 Small are MobileNetV3, and V2 are standard MobileNetV2. All metrics were measured on single shot use of the same device using TensorFlow. MobileNetV3 Large and Small are our proposed next generation mobile models.

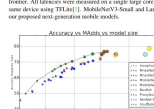


Figure 2: The trade-off between MAs and top-1 accuracy. This allows to compare models that were targeted different hardware or software frameworks. All MobileNetV3 are for input resolution 224 and use resolution 13, 15, 17, 19, 21 and 23. See section 6 for more details. See version 3.0 for more details.

arXiv:1905.02244v5 [cs.CV] 20 Nov 2019

1. Introduction

Efficient neural networks are becoming ubiquitous in mobile applications enabling critical new on-device experience. They are also a key enabler of personal privacy as being able to gain the benefit of neural networks without needing to send their data to the server to be evaluated. Advances in neural network efficiency not only improve user experience via higher accuracy and lower latency, but also

help preserve battery life through reduced power consumption.

This paper describes the approach we took to design MobileNetV3 Large and Small models in order to deliver the next generation of high accuracy efficient mobile work models to power on device computer vision. The new networks push the state of the art forward and demonstrate how to build automated search with neural architecture advances to build efficient models.

arXiv:1801.04381v4 [cs.CV] 21 Mar 2019

MobileNetV3 Small

Convolutacional 3x3



Bottleneck 3x3



Bottleneck 5x5



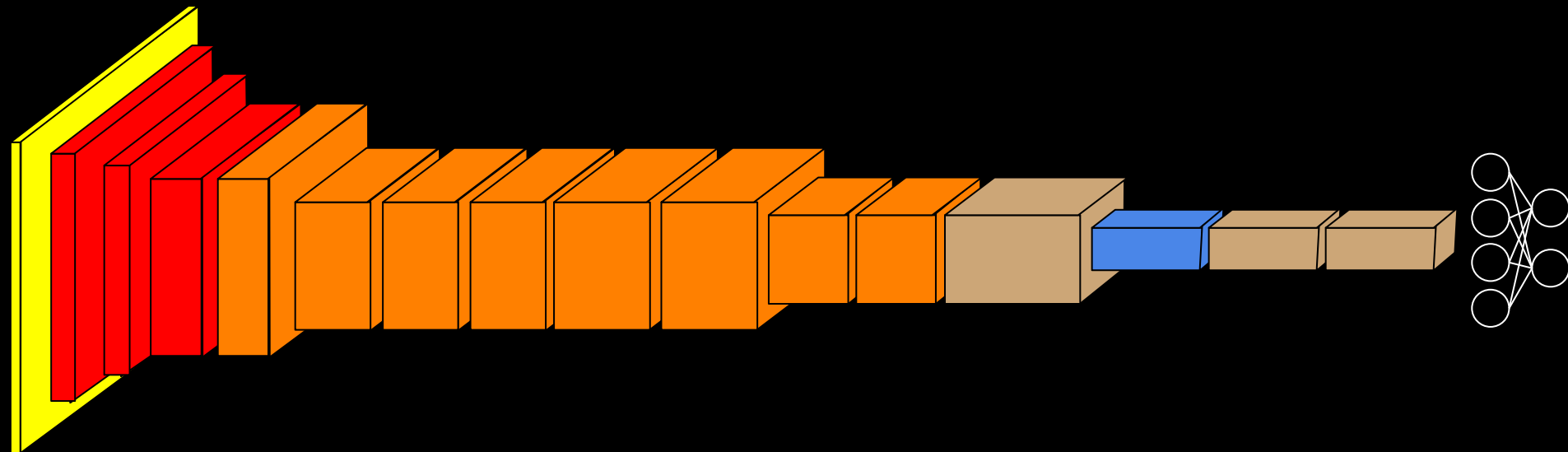
Pooling



Convolutacional 1x1



Tensores de entrada 224x224x3.



MobileNetV3 Small

Convolutacional 3x3



Bottleneck 3x3

Bottleneck 5x5

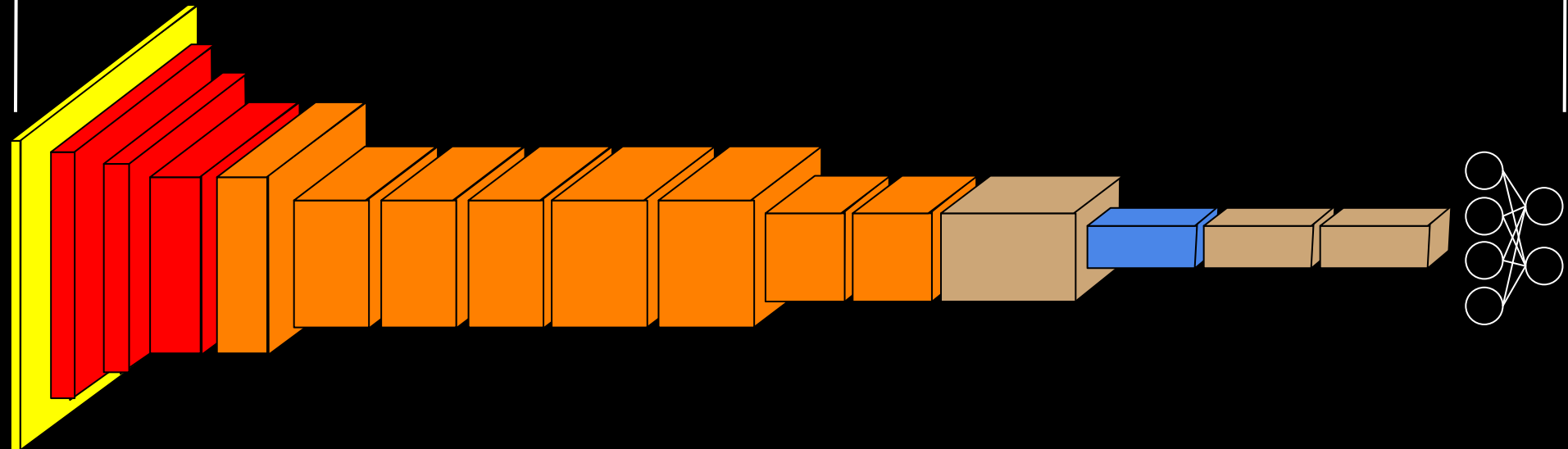


Pooling

Convolutacional 1x1



Treinar a rede completa na ImageNet - aprox. 14 milhões de imagens.



MobileNetV3 Small

Convolutacional 3x3



Bottleneck 3x3



Bottleneck 5x5



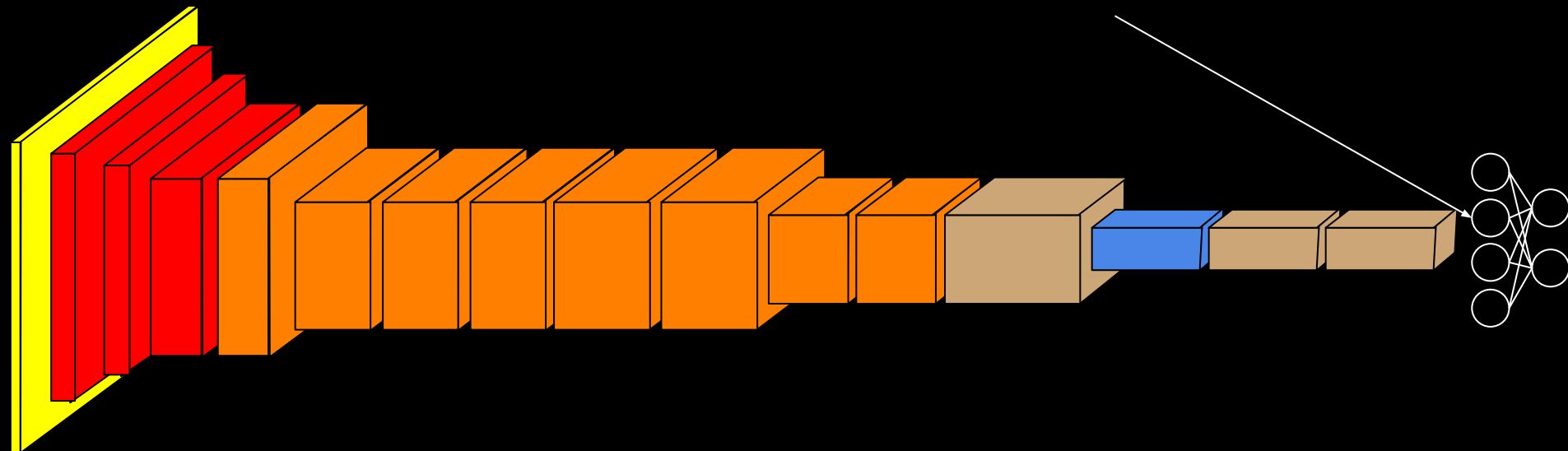
Pooling



Convolutacional 1x1



Criar uma nova camada fully-connected, ou retreinar a camada existente nos dados alvo. Não alterar os pesos das demais camadas.



MobileNetV3 Small

Convolutacional 3x3



Bottleneck 3x3



Bottleneck 5x5



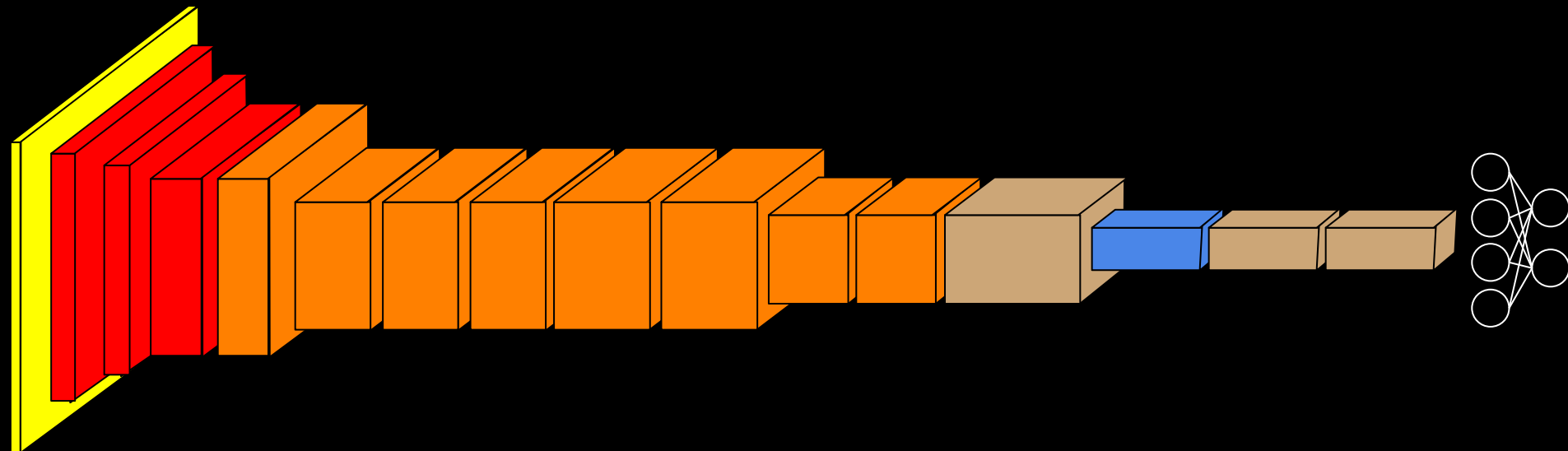
Pooling



Convolutacional 1x1



Camadas que não alteramos os pesos geralmente são chamadas de **camadas congeladas**.



Faça você mesmo

Execute o exemplo de Transfer Learning no Google Colab disponibilizado.

Fine-tuning

Um fine-tuning de todas as camadas da rede pode ser considerado uma espécie de inicialização da rede.

A rede foi inicializada em uma região do espaço “boa para outro problema”.

Esperamos que essa região seja, pelo menos, próxima de uma boa região para o problema em questão.

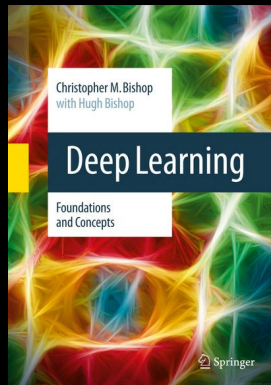
Geralmente usamos fatores de aprendizagem baixos para realizar o fine-tuning.

Exercícios

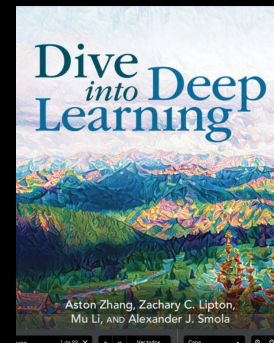
1. Baseado no exemplo de transfer learning.
 - a. Faça um Fine-tuning em todas as camadas da rede.
 - b. Treine, e compare os resultados com os obtidos com o transfer learning.

Referências

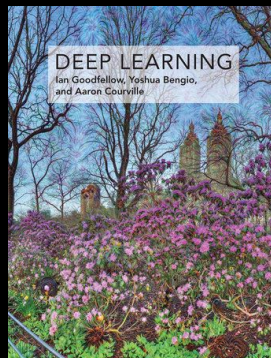
Bishop, C. M., Bishop, H. Deep Learning: Foundations and Concepts. 2023.



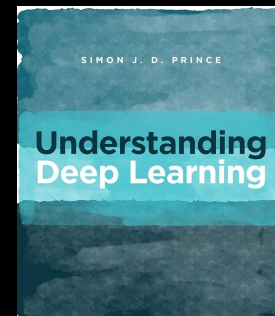
Zhang, A., Lipton, Z. C., Li, M., Smola, A. J. Dive Into Deep Learning. Índia. 2023.



Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. 2016.



Prince, S. J. Understanding Deep Learning. 2023.



Licença

Esta obra está licenciada com uma Licença [Creative Commons Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).