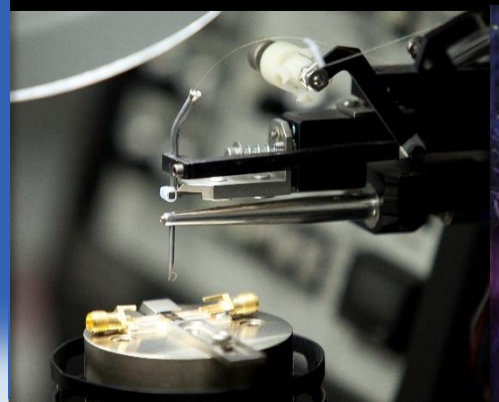
Centro Brasileiro de Pesquisas Físicas

Redes Neurais profundas e aplicações Deep Learning

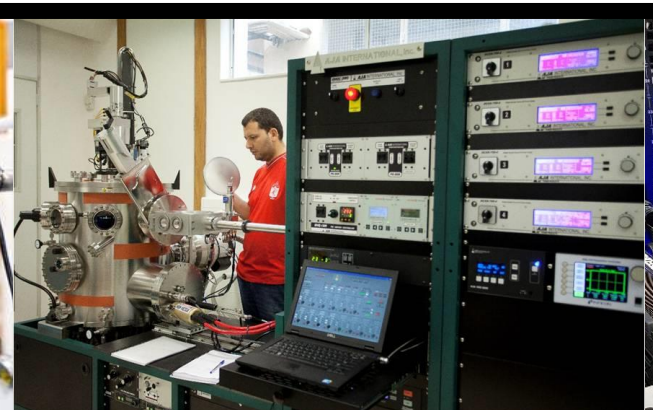
Clécio Roque De Bom – debom@cbpf.br

CBPF

clearnightsrthebest.com








Going Deeper with Convolutions - AKA - Inception

Bigger size typically means a larger number of parameters, which makes the enlarged network more prone to overfitting, especially if the number of labeled examples in the training set is limited. (…)The other drawback of uniformly increased network size is the dramatically increased use of computational resources. For example, in a deep vision network, if two convolutional layers are chained, any uniform increase in the number of their filters results in a **quadratic increase of computation**. If the added capacity is used inefficiently (for example, **if most weights end up to be close to zero), then much of the computation is wasted**.(…).

storage.googleapis.com/pub-tools-public-publication-data/pdf/43022.pdf

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Going Deeper with Convolutions - AKA - Inception

A fundamental way of solving both of these issues would be **to introduce sparsity and replace the fully connected layers by the sparse ones, even inside the convolutions.** Besides mimicking biological systems, this would also have the advantage of firmer theoretical underpinnings due to the groundbreaking work of Arora et al. [2]. **Their main result states that if the probability distribution of the dataset is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer after layer by analyzing the correlation statistics of the preceding layer activations and clustering neurons with highly correlated outputs.**

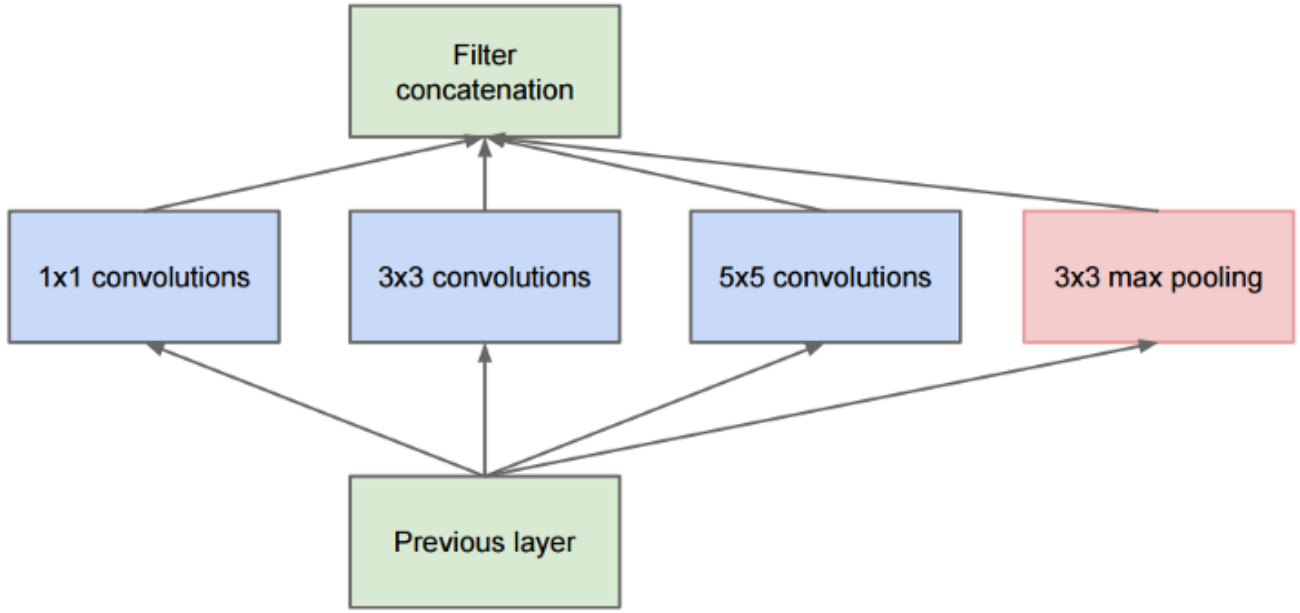
storage.googleapis.com/pub-tools-public-publication-data/pdf/43022.pdf

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

The Main (Inception) Idea...

GoogLeNet (2014)

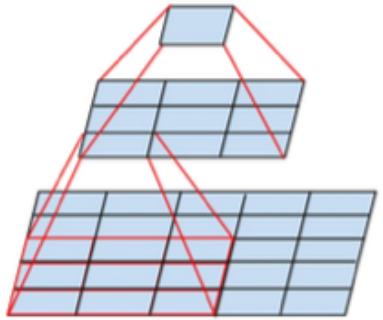
The idea is that you don't need to know in advance if it was better to do, for example, a 3×3 then a 5×5. Instead, just do all the convolutions and let the model pick what's best. Additionally, this architecture allows the model to recover both local feature via smaller convolutions and high abstracted features with larger convolutions.



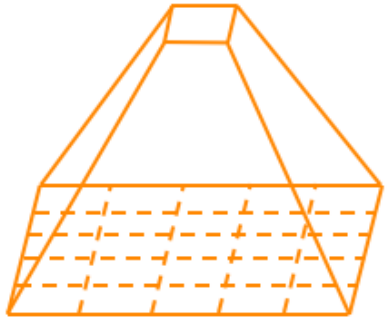
The Main (Inception) Idea...

GoogLeNet (2014)

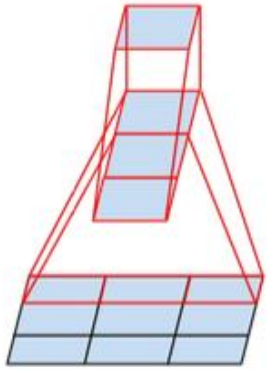
5 million (V1) and 23 million (V3)



two successive
3x3 convolutions



5x5 convolution



3x3 convolutions could be further deconstructed into successive 3x1 and 1x3 convolutions.

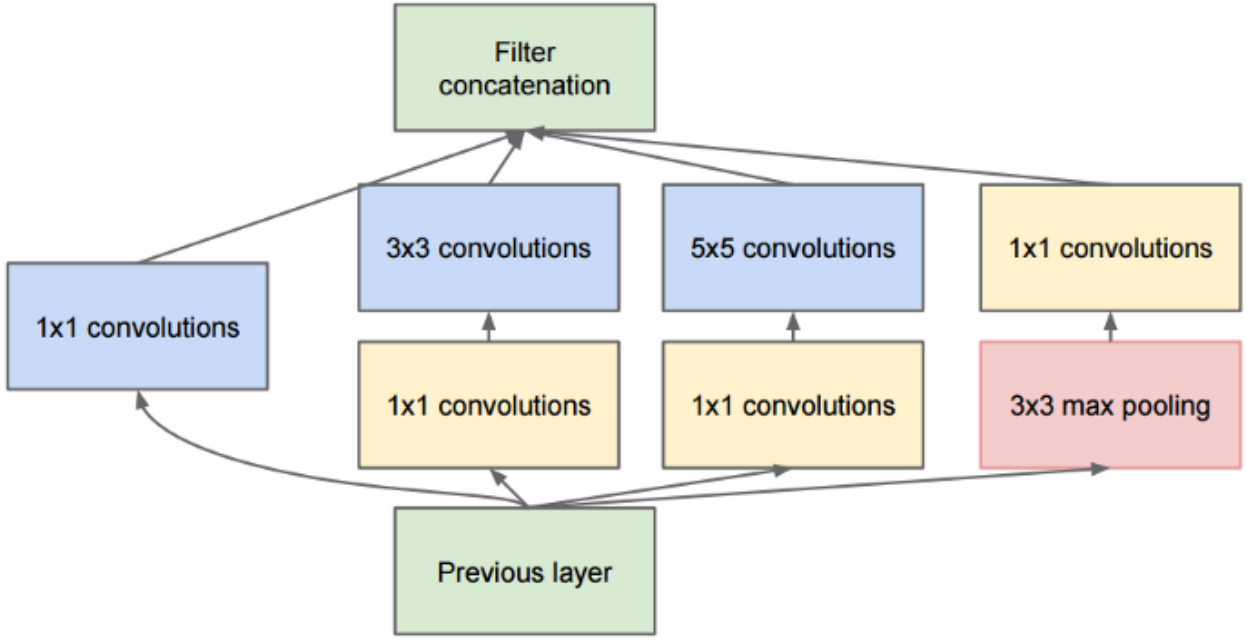
Generalizing this insight, we can more efficiently compute an n×n convolution as a 1×n convolution followed by a n×1 convolution..

https://www.jeremyjordan.me/convnet-architectures/

What to choose?

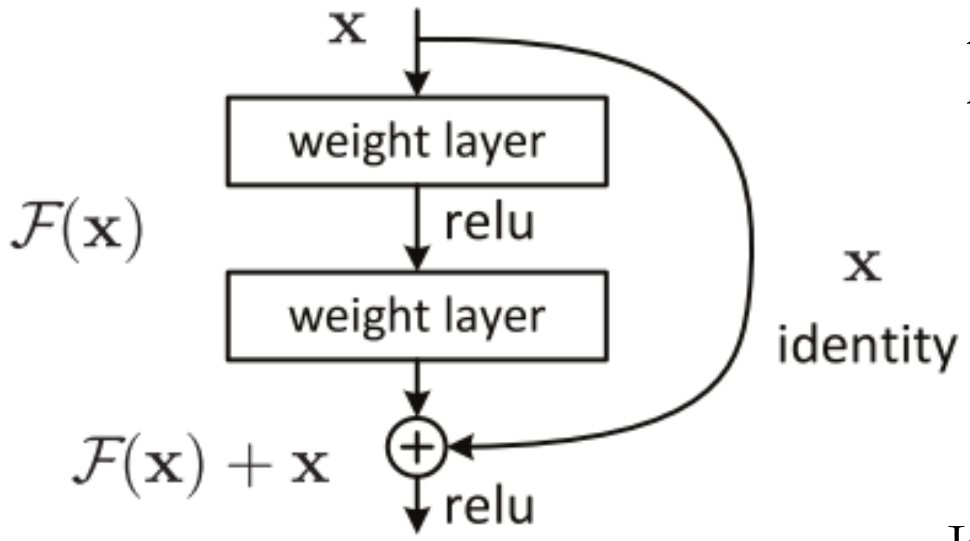
Merge layers

Add/Merge (+) Subtract (-) Multiply (*) Average (~) Maximum (V) Concatenate (⌒) Dot (×)



We still have millions of parameters to fit!!!! We still need some ideas to prevent overfitting

ResNet Block



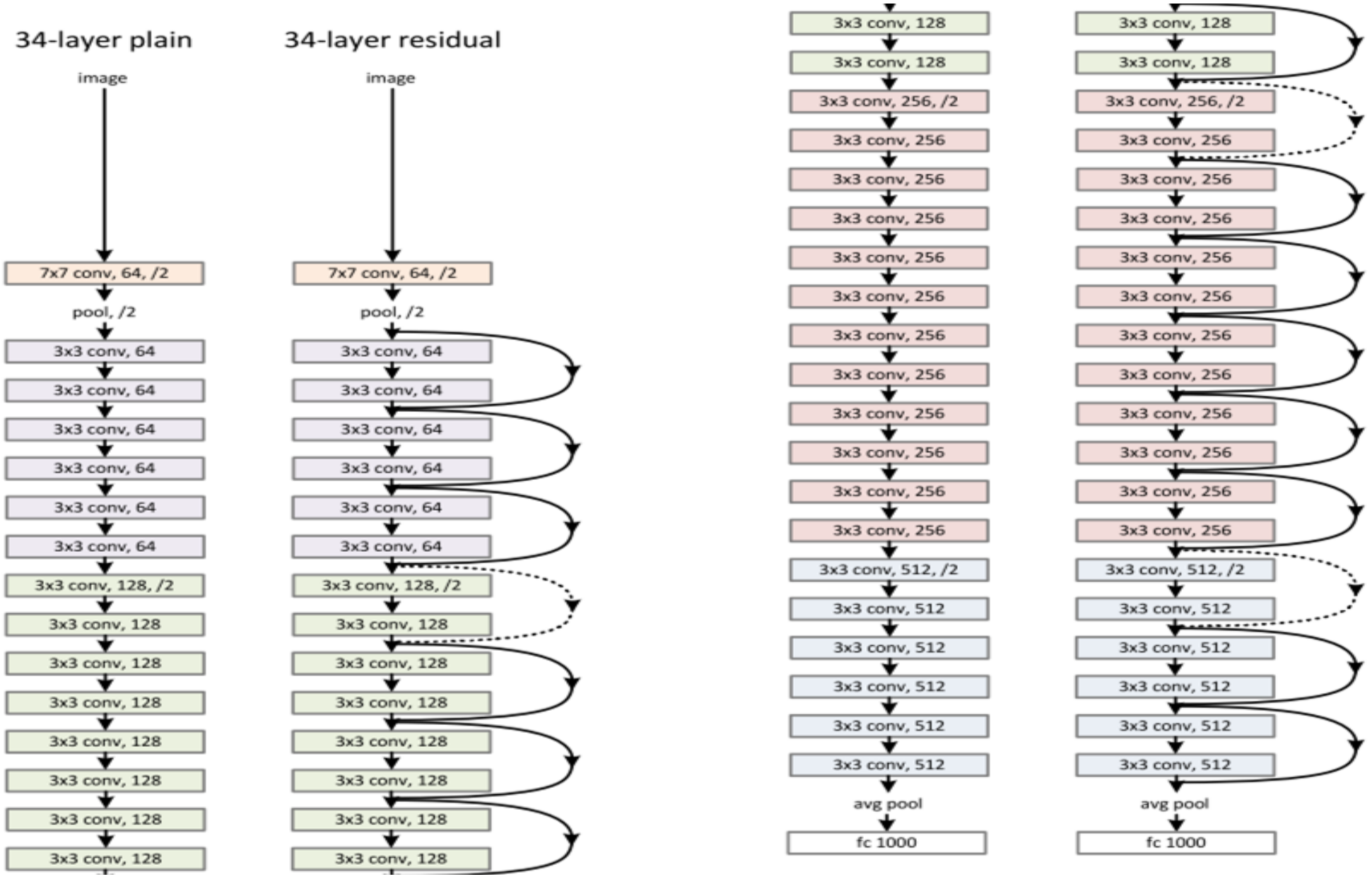
The author's hypothesis is that it is easy to optimize the residual mapping function F(x) than to optimize the original, unreferenced mapping .

$H(x) = F(x)+x$

If the identity mapping is optimal, We can easily push the residuals to zero (F(x) = 0) than to fit an identity mapping (x, input=output) by a stack of non-linear layers.

It also put a new light on the vanishing gradient problem...

Residual Neural Networks



Residual Neural Networks

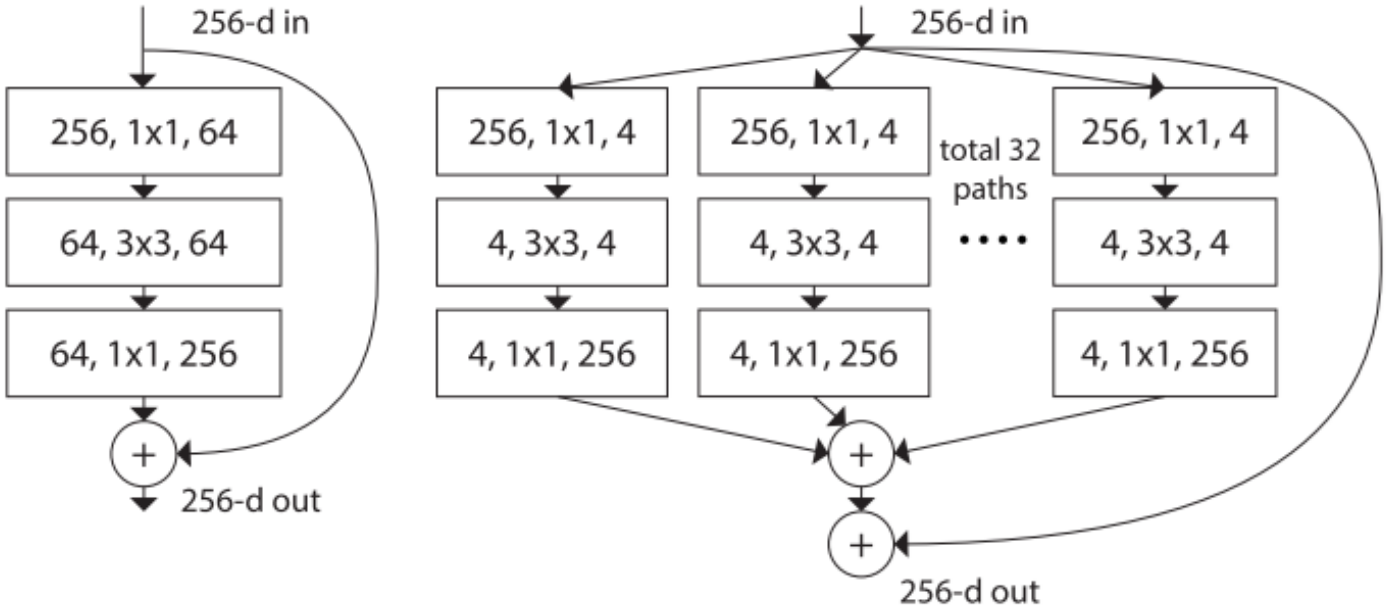
- Won 1st place in the ILSVRC 2015 classification competition with top-5 error rate of 3.57% (An ensemble model)
- Won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation.
- Replacing VGG-16 layers in Faster R-CNN with ResNet-101. They observed a relative improvements of 28%
- Efficiently trained networks with 100 layers and 1000 layers also.
- ResNet Network Converges faster compared to plain counterpart of it.
- Identity vs Projection shortcuts. Very small incremental gains using projection shortcuts in all the layers. So all ResNet blocks use only Identity shortcuts with Projections shortcuts used only when the dimensions changes.

Residual Neural Networks

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Resnext

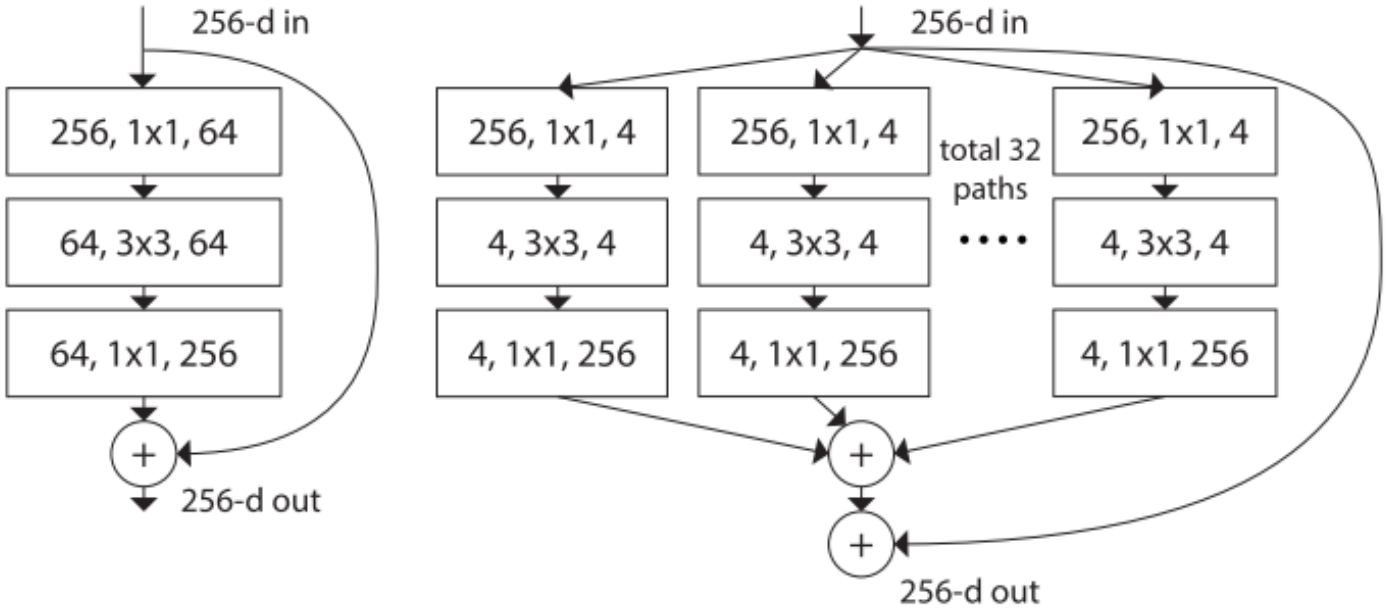


- Similar to Inception. However, the outputs of different paths are merged by adding them together, in the Inception they are concatenated.

- In the Inception paper each path is different (1x1, 3x3 and 5x5 convolution) and in the ResNext all paths share the same topology.

S. Xie, R. Girshick, P. Dollar, Z. Tu and K. He. Aggregated Residual Transformations for Deep Neural Networks. arXiv preprint arXiv:1611.05431v1,2016.

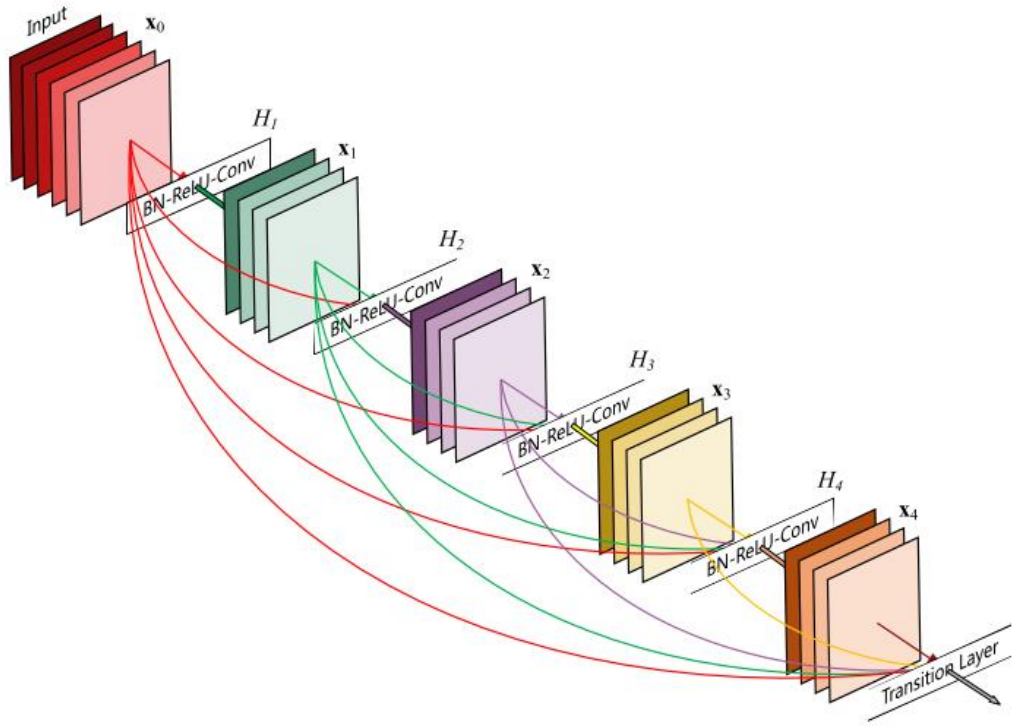
Resnext



- Cardinality: the number of independent paths, to provide a new way of adjusting the model capacity.
- Experiments show that accuracy can be gained more efficiently by increasing the cardinality than by going deeper or wider.
- Less Hyper-parameters than Inception

S. Xie, R. Girshick, P. Dollar, Z. Tu and K. He. Aggregated Residual Transformations for Deep Neural Networks. arXiv preprint arXiv:1611.05431v1,2016.

Densely Connected Convolutional Networks



Huang et al. argue that this architecture encourages feature reuse, making the network highly parameter-efficient.

Concatenating feature maps can preserve them all and increase the variance of the outputs.

G. Huang, Z. Liu, K. Q. Weinberger and L. Maaten. Densely Connected Convolutional Networks. arXiv:1608.06993v3,2016.

How to Implement it?

You can access a layer's regularization penalties by calling `layer.losses` after calling the layer on inputs:

```
layer = tf.keras.layers.Dense(5, kernel_initializer='ones',
                              kernel_regularizer=tf.keras.regularizers.l1(0.01),
                              activity_regularizer=tf.keras.regularizers.l2(0.01))
tensor = tf.ones(shape=(5, 5)) * 2.0
out = layer(tensor)
# The kernel regularization term is 0.25
# The activity regularization term (after dividing by the batch size) is 5
print(tf.math.reduce_sum(layer.losses))  # 5.25 (= 5 + 0.25)
```

Available regularizers

The following built-in regularizers are available as part of the `tf.keras.regularizers` module:

`L1` class

```
tf.keras.regularizers.l1(l1=0.01, **kwargs)
```

A regularizer that applies a L1 regularization penalty.

The L1 regularization penalty is computed as: `loss = l1 * reduce_sum(abs(x))`

L1 may be passed to a layer as a string identifier:

```
>>> dense = tf.keras.layers.Dense(3, kernel_regularizer='l1')
```

In this case, the default value used is `l1=0.01`.

https://keras.io/api/layers/regularizers/

How to Implement a ResNet ?

```
import numpy as np
import pandas as pd
import tensorflow as tf
import h5py
import os
#from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
```

```
res_net = tf.keras.applications.ResNet50(input_shape = (X.shape[1:]),
include_top = False, weights='imagenet')
flat = tf.keras.layers.Flatten()(res_net.output)
y_hat = tf.keras.layers.Dense(1, activation = "sigmoid")(flat)
model = tf.keras.models.Model(res_net.input, y_hat)
```

https://keras.io/api/applications/

How to Implement a ResNet ?

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

https://keras.io/api/applications/

How to Implement Extract Features?

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np

model = VGG16(weights='imagenet', include_top=False)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

How to Implement Extract Features?

```
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_input
from tensorflow.keras.models import Model
import numpy as np

base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('block4_pool').output)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

block4_pool_features = model.predict(x)
```

Centro Brasileiro de Pesquisas Físicas

Redes Neurais profundas e aplicações Deep Learning

Clécio Roque De Bom – debom@cbpf.br

CBPF

clearnightsrthebest.com