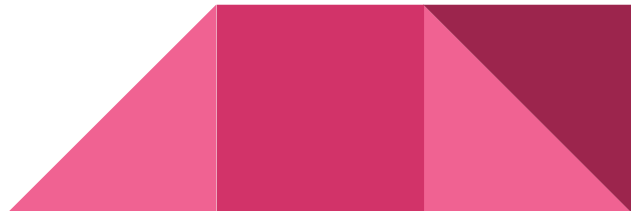


TCC Lucas G. Vinhas

CLASSIFICAÇÃO DE OBRAS DE ARTE UTILIZANDO
APRENDIZADO DE MÁQUINA.

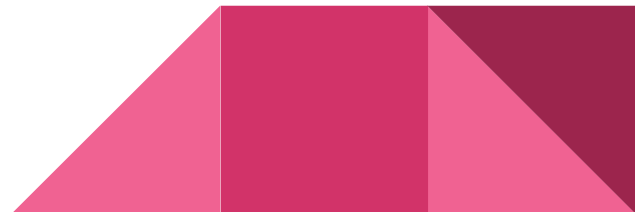
Introdução

- A ideia geral é classificar obras de arte com o uso de Aprendizado de máquina.
- São catalogadas manualmente por curadores de acordo com suas escolas artísticas, como Classicismo, Modernismo, Impressionismo etc. Só na década de 80 que surgiram projetos para utilizar computação para ajudar na catalogação.
- Estudo recentes mostraram que Aprendizado de Máquina é uma abordagem muito eficaz para problemas de classificação.



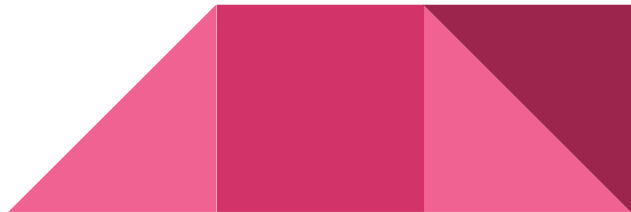
Desafios

- Existem muitas características semelhantes em categorias diferentes.
- É difícil encontrar uma arquitetura baseada em CNN satisfatória, pois as características extraídas entre as camadas inferiores e superiores podem ser muito diferentes na mesma categoria.



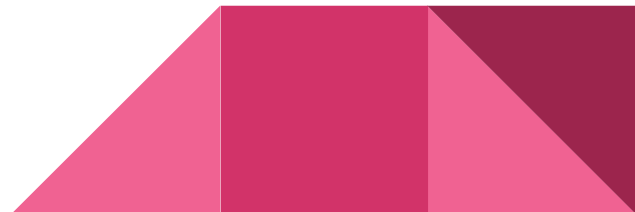
Objetivos

- Investigar e viabilizar diferentes arquiteturas baseadas em Redes Neurais de Convolução (CNN).
- Explorar o uso da técnica transferência de aprendizado, do inglês Transfer Learning, na tarefa de classificar pinturas de acordo com seus respectivos artistas.
- Contribuir com mais estudos na área proposta.

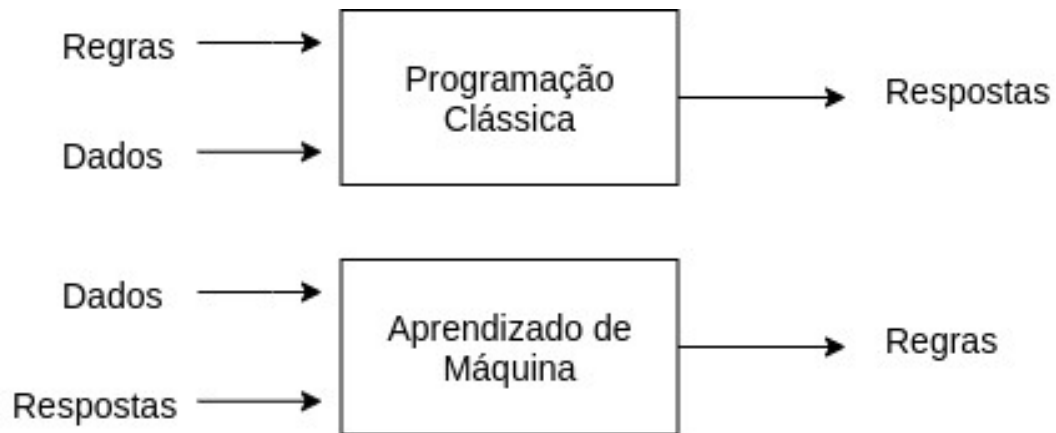


Fundamentação Teórica

- Aprendizado de Máquina
- Reconhecimento de Padrões
- Redes Neurais Artificiais
- Convolution Neural Network (CNN)

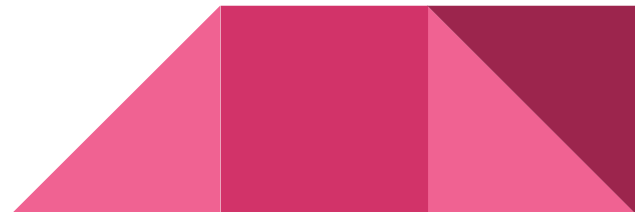


Aprendizado de Máquina



Reconhecimento de Padrões

- Verdadeiro Positivo: O modelo previu X e a resposta é X;
- Verdadeiro Negativo: O modelo previu que não é X e a resposta não é X;
- Falso Positivo: O modelo previu X e a resposta não é X;
- Falso Negativo: O modelo previu que não é X e a resposta é X.



Critérios

- Acurácia

$$A = \frac{\sum \text{Verdadeiros Positivos} + \sum \text{Verdadeiros Negativos}}{\text{Numero de amostras Teste}}$$

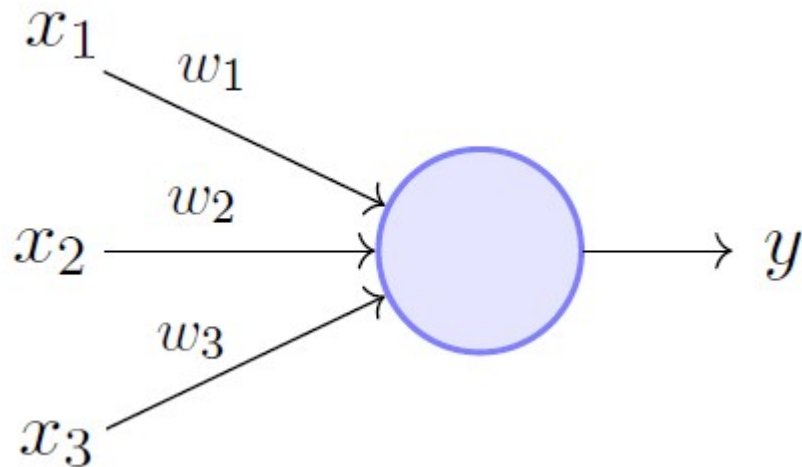
- Precisão

$$P = \frac{\sum \text{Verdadeiros Positivos}}{\sum \text{Verdadeiros Positivos} + \sum \text{Falsos Positivos}}$$

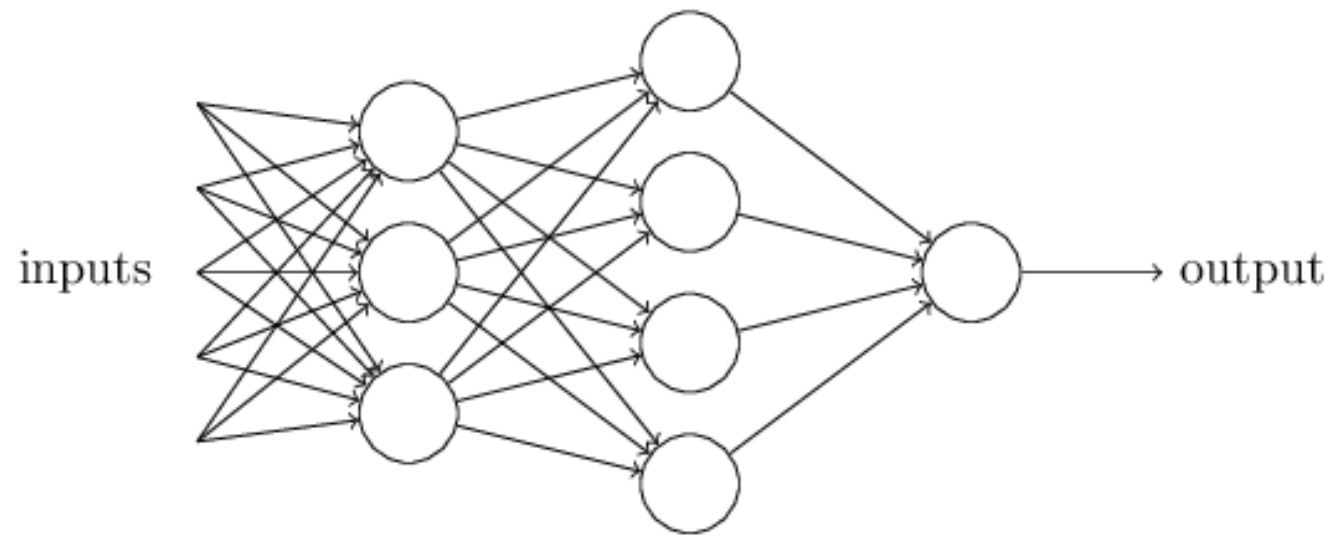
- Recal

$$R = \frac{\sum \text{Verdadeiros Positivos}}{\sum \text{Verdadeiros Positivos} + \sum \text{Falsos Negativos}}$$

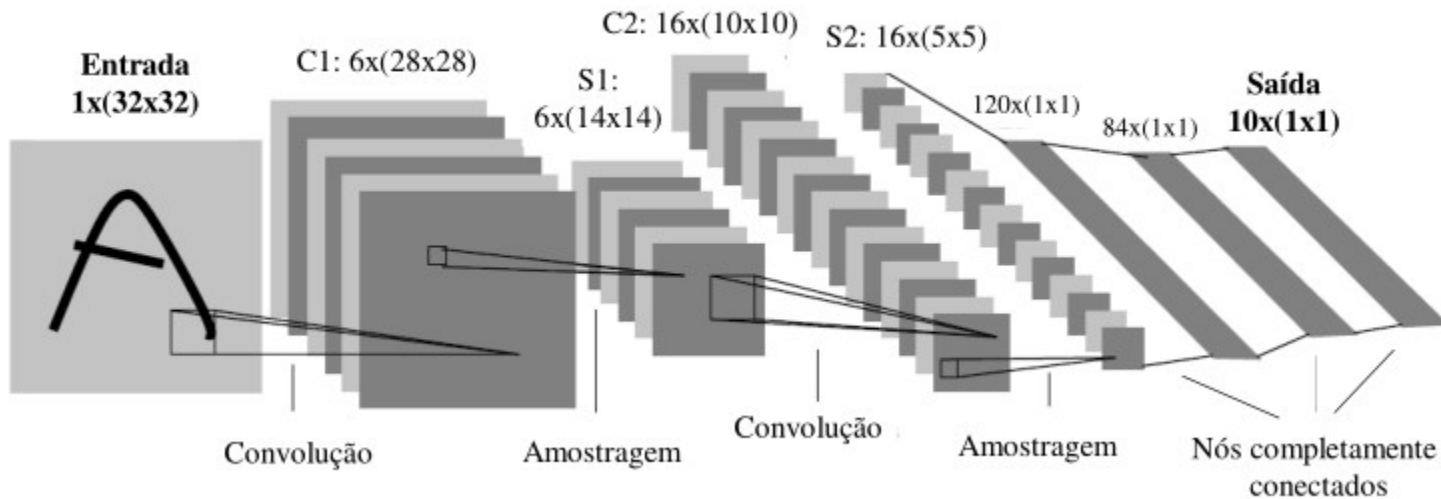
Redes Neurais Artificiais



$$saída = \begin{cases} 0 & \text{se } \sum_j w_j x_j \leq \text{limiar} \\ 1 & \text{se } \sum_j w_j x_j \geq \text{limiar} \end{cases}$$

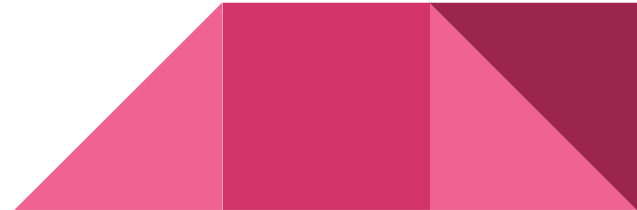


CNN - Convolutional Neural Network

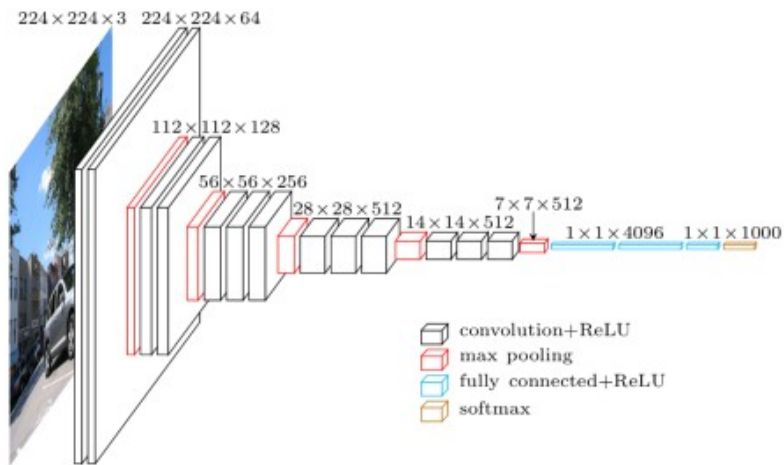


Arquiteturas baseadas em CNN

- VGG
- Resnet
- Inception
- InceptionResnet
- Xception



Vgg16



Rede de 2015

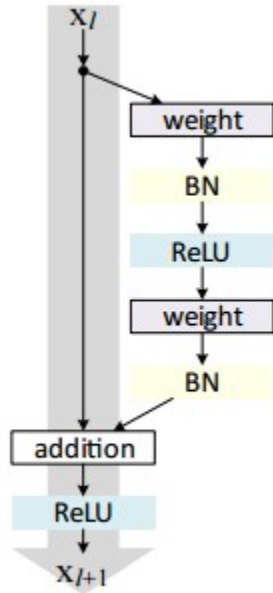
Tamanho de imagem padrão: 224 X 224.

16 camadas de treino com peso. Nã época, 16 camadas era muito.

O grande problema é o tamanho do modelo.

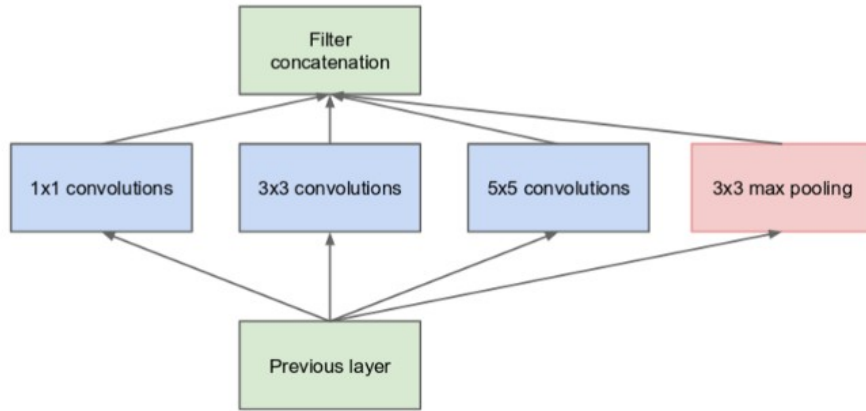
De todas as redes usada, é a que possui mais parâmetros.

Resnet



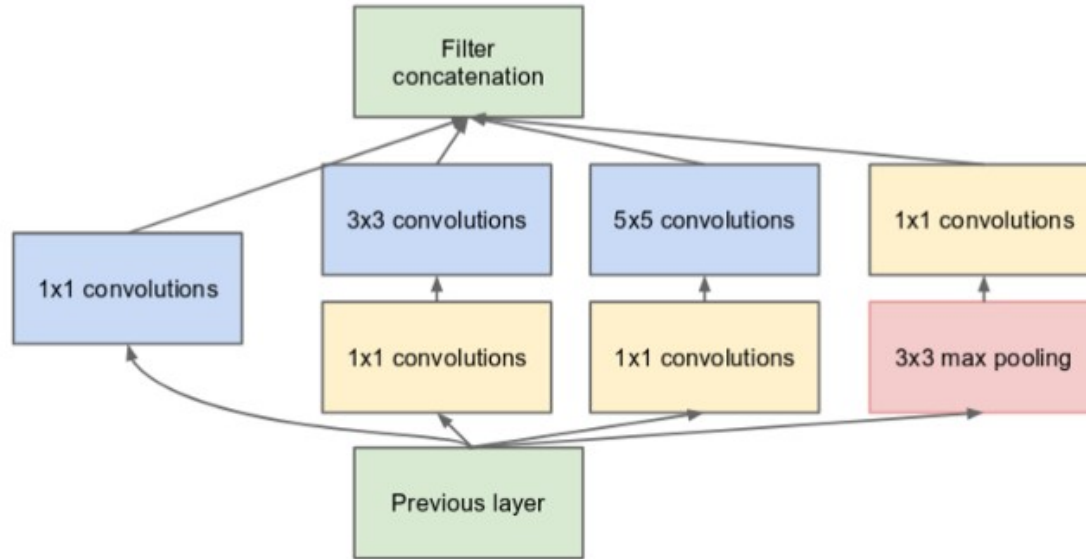
- Nome derivado de “residual network”.
- Rede permite o treinamento de forma mais rápida do que as demais, mesmo permitindo o uso de muitas camadas.
- Possui arquiteturas com variado número de camadas.
- Baseadas nas células piramidais do córtex humano.
- Elas conseguem “pular” camadas durante o treinamento para otimizar seus resultados.
- Tenta resolver o problema gerado por redes de muitas camadas.

Inception



(a) Inception module, naïve version

- Desenvolvida pela equipe de inteligência do Google
- Avanço na forma de criar arquiteturas

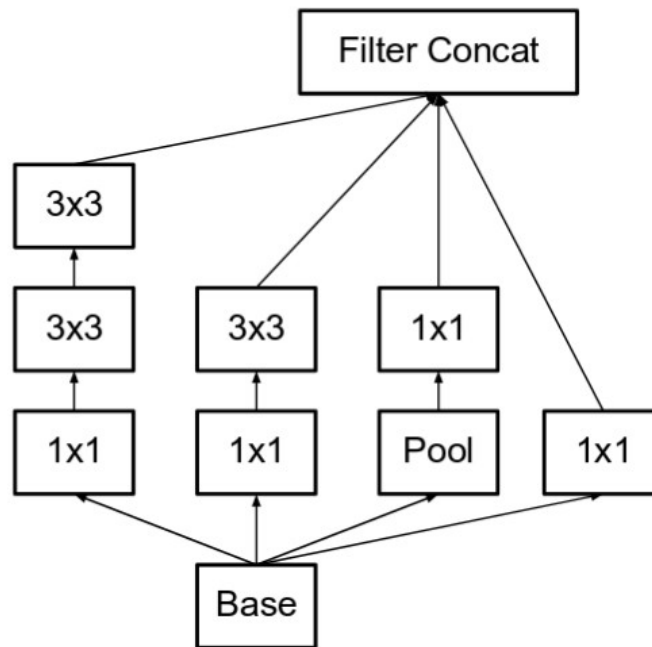


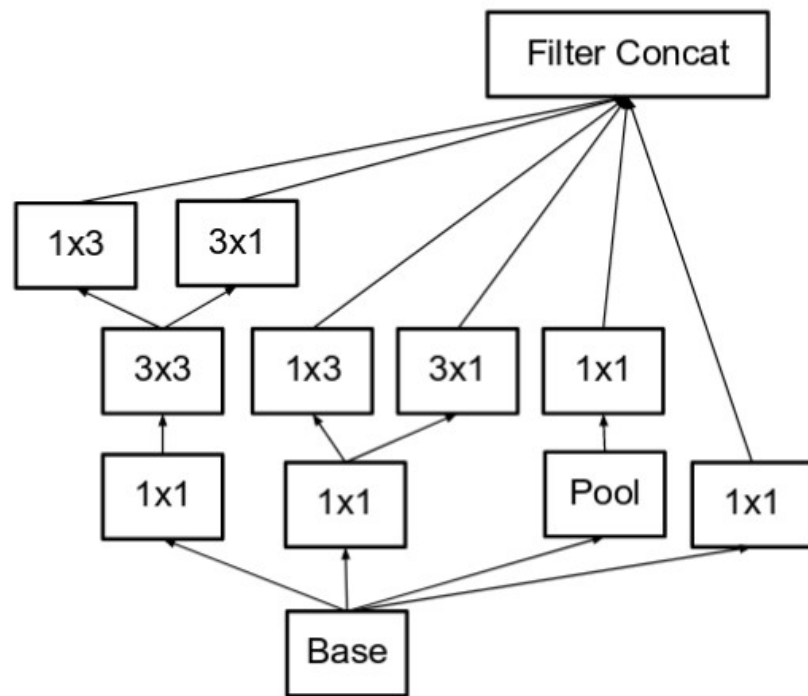
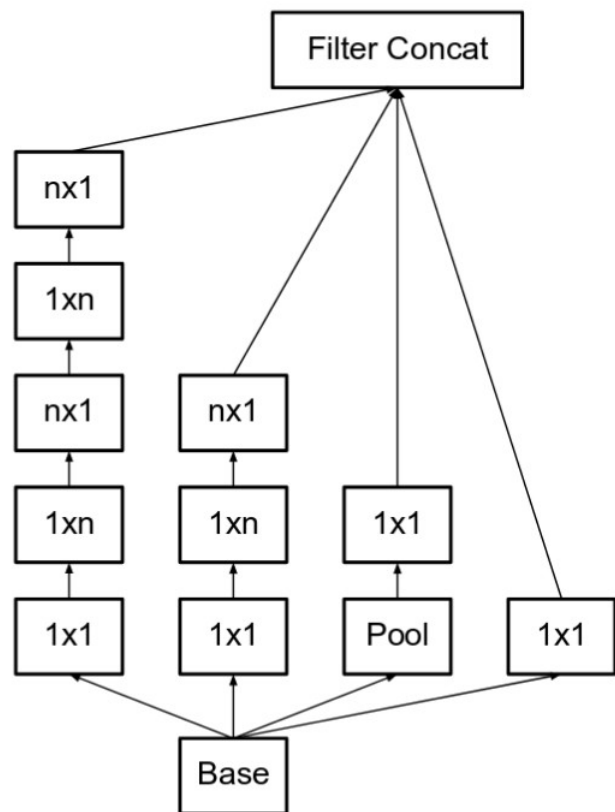
(b) Inception module with dimension reductions

InceptionV2

- Fatorações para performance

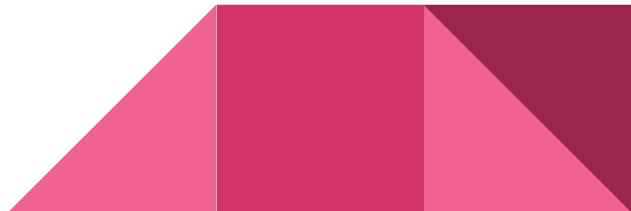
- Tamanho final da rede



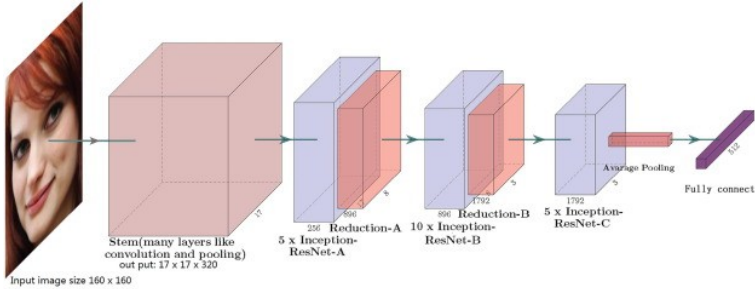


InceptionV3

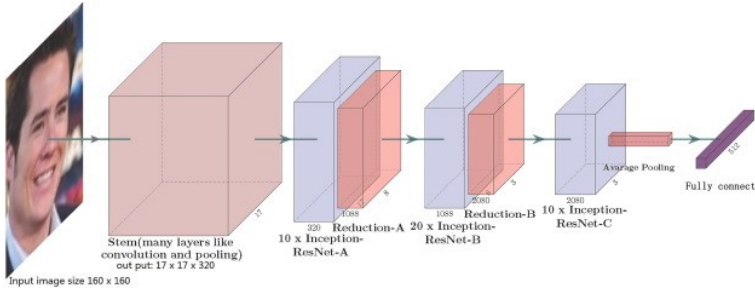
- Usar o RMSProp como otimizador.
- Adicionar convoluções fatoráveis de 7×7 .
- Adicionar normalização de lote nos classificadores auxiliares.
- Adicionar um regularizador na fórmula de perda para impedir a rede de atingir overfitting.



InceptionResnetV2



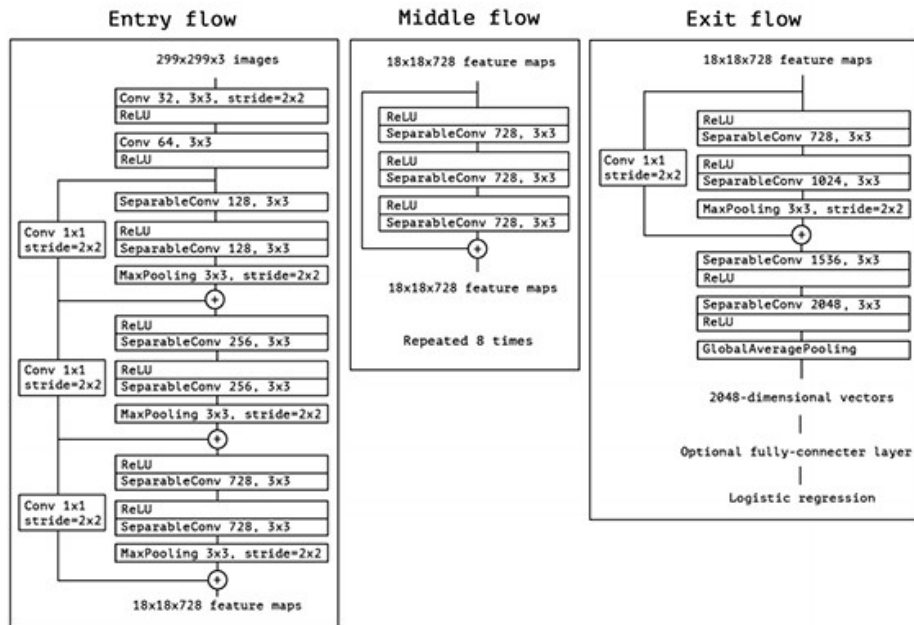
(a) architecture of Inception-ResNet v1



(b) architecture of Inception-ResNet v2

- A ideia foi combinar a eficácia da Inception com a performance da Resnet.
- Tamanho padrão de imagens é (299 X 299)
- Paper de 2017

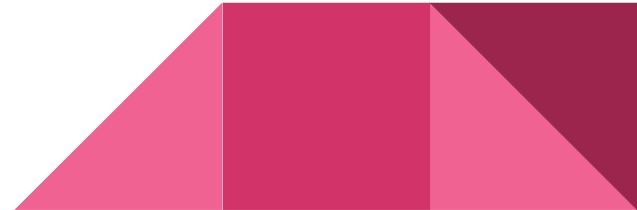
Xception



- Feita pelo criador do Keras.
- Substituiu os inceptions modules para usar “Depthwise separable convolutions”.
- Possui 126 camadas.

Ferramentas

- Python.
- Tensorflow.
- Kaggle Notebooks.
- Google Colabs.
- Jupyter Notebooks.
- Cuda.
- Keras.



Kaggle notebooks

The screenshot displays the Kaggle notebook interface for a notebook titled 'notebook328a8a0435'. The interface includes a top bar with 'File', 'Edit', 'View', 'Run', 'Add-ons', and 'Help' menus. A 'Share' button and a 'Save Version' button (showing 0 versions) are also present. The main area contains a code cell with the following content:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

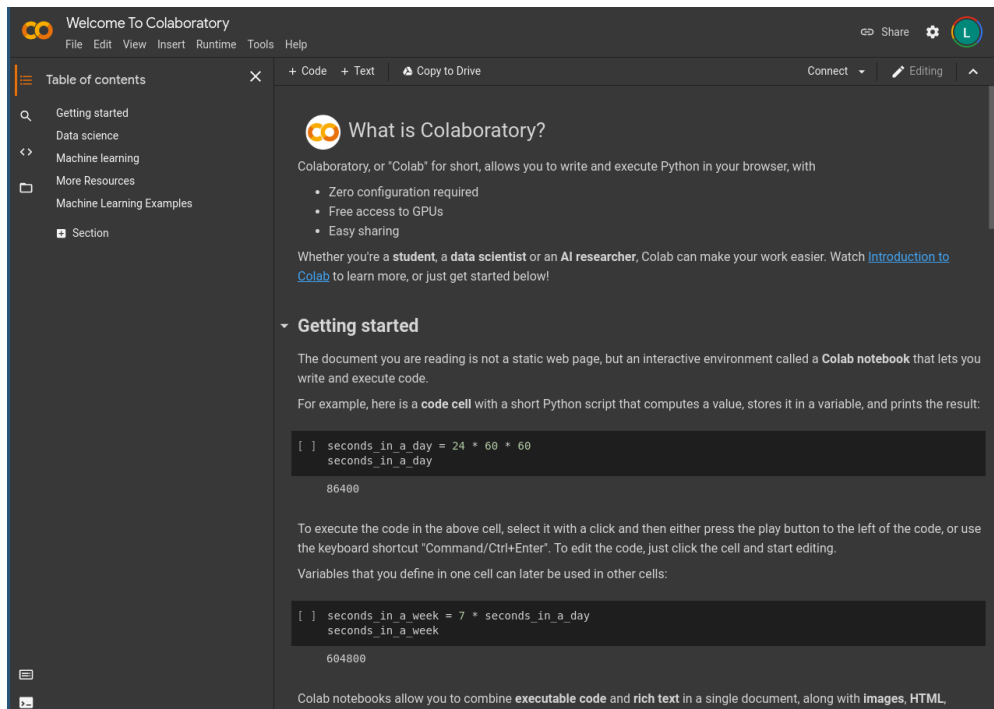
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the
```

Below the code cell are buttons for '+ Code' and '+ Markdown'. The right sidebar shows the 'Data' section with 'input' and 'output' folders, and the 'Settings' section with options for Language (Python), Environment (Preferences), Accelerator (None), and Internet (checked).

Gpu: 40 horas semanais

Google colab



The screenshot displays the Google Colaboratory web interface. The top navigation bar includes the Colab logo, a 'Welcome To Colaboratory' message, and a menu with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right of the top bar are links for 'Share', a settings gear, and a user profile icon. A left sidebar contains a 'Table of contents' with links to 'Getting started', 'Data science', 'Machine learning', 'More Resources', and 'Machine Learning Examples'. The main content area is titled 'What is Colaboratory?' and explains that Colab allows writing and executing Python in a browser. It lists three benefits: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. It also mentions that Colab is suitable for students, data scientists, and AI researchers. A section titled 'Getting started' describes the Colab notebook as an interactive environment. It provides an example of a code cell that calculates the number of seconds in a day (24 * 60 * 60), showing the output '86400'. It also explains how to execute code (via a play button or keyboard shortcut) and how to edit code (by clicking the cell). A second code cell is shown calculating the number of seconds in a week (7 * seconds_in_a_day), with the output '604800'. The bottom of the page states that Colab notebooks allow combining executable code, rich text, images, and HTML.

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

Share Settings User

Table of contents

- Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples

Section

What is Colaboratory?

Colaboratory, or 'Colab' for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut 'Command/Ctrl+Enter'. To edit the code, just click the cell and start editing.

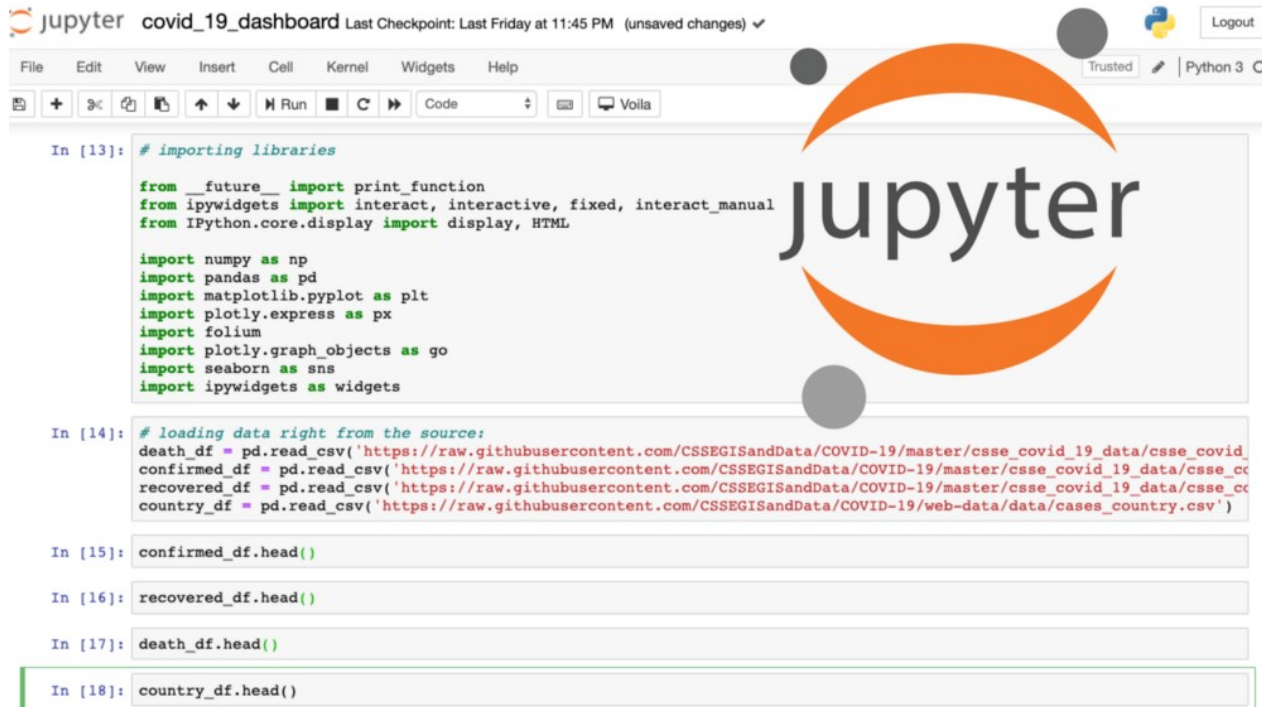
Variables that you define in one cell can later be used in other cells:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week
```

604800

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**,

Jupyter Notebooks



```
In [13]: # importing libraries

from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
from IPython.core.display import display, HTML

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import folium
import plotly.graph_objects as go
import seaborn as sns
import ipywidgets as widgets

In [14]: # loading data right from the source:
death_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/country_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv')

In [15]: confirmed_df.head()

In [16]: recovered_df.head()

In [17]: death_df.head()

In [18]: country_df.head()
```

Tensor Flow

An end-to-end open source
machine learning platform

É um conjunto de ferramentas,
bibliotecas e recursos, criados para
facilitar o uso de Machine Learning
para Desenvolvedores

criada pelo google

Utiliza python como linguagem padrão,
mas tem suporte a outras linguagens.

Recomendado tanto para projetos
acadêmicos como para projetos
corporativos.

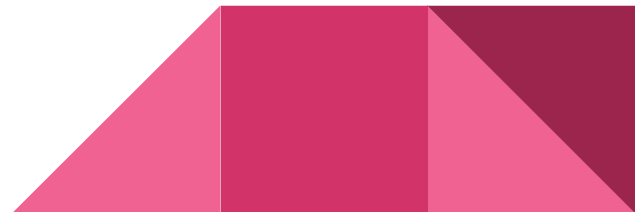


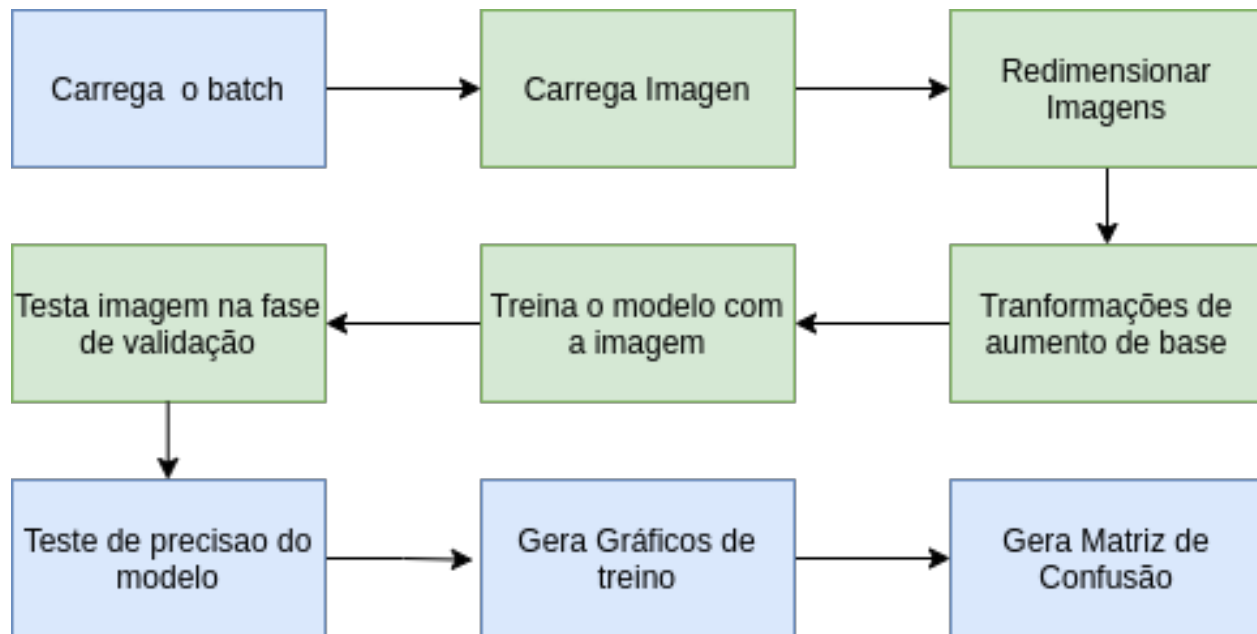
Keras



Pipeline do Tensorflow

- 1) Primeiro divide as pastas em train , test , validation , 70%-15%-15%.
- 2) Transforma a imagem para o tamanho desejado.
- 3) Aplica data Augmentation na imagem.
- 4) treinar o modelo com o a imagem.



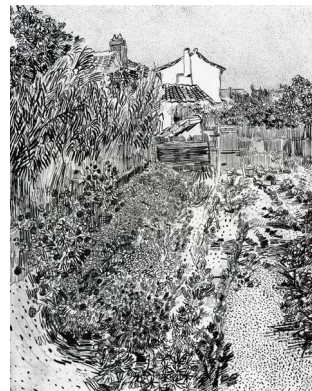
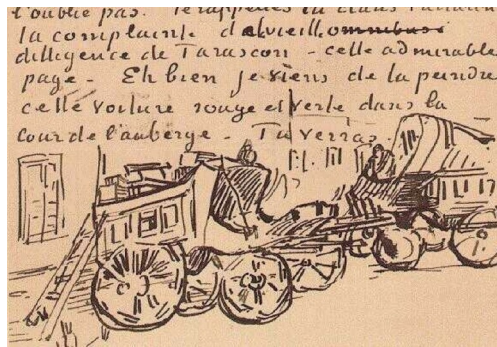


Base utilizada - Best Artworks of All Time

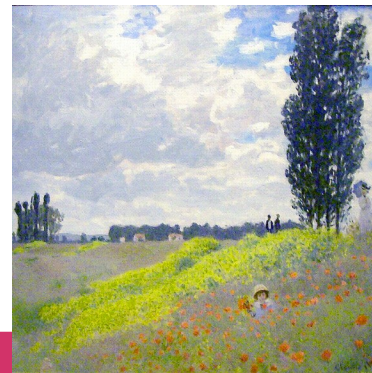
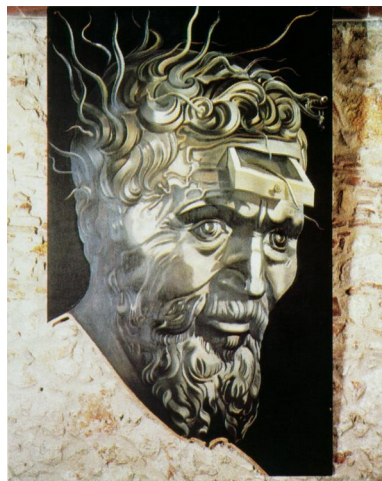
- Base de dados composta por 50 diferentes artistas.
- Todos os artistas , possuem 8 diferentes características para classificação: o nome, os anos de atividade, o gênero, a sua nacionalidade, uma breve biografia, sua página no wikipedia e diversas pinturas deles.
- Originalmente proposta para adivinhar quem é o pintor de uma dada pintura. Isso usando as cores e os padrões geométricos .
- É dividida em três partes: um arquivo com todas as características e informações de cada artista , uma parte com as imagens originais e outra parte com as imagens redimensionadas.



Exemplo: Van Gogh

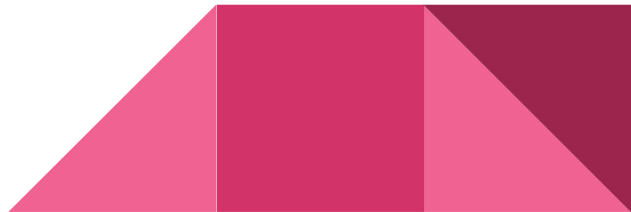


Exemplos



Algoritmo

- 50 épocas iniciais.
- O inicia com o learning rate 1×10^{-4} , depois de 5 épocas se melhora ele diminui em 10^{-1} .



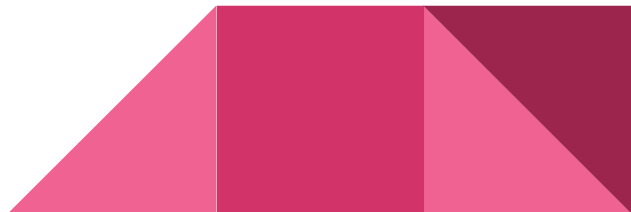
Data Augmentation

- Técnica utilizada quando o tamanho do dataset não é suficientemente grande para resolver o problema, causando overfitting muito rápido e gerando uma taxa de precisão muito baixa.
- A ideia central é mostrar a mesma foto de uma forma diferente. A mesma foto é treinada ao contrário, com zoom, pela metade etc. A ideia é conseguir obter melhores resultados sem aumentar o tamanho real do dataset.
- Os tipos de aumento de dados utilizados foram: mudança de escala, rotação, distorção no eixo x e inversão nos eixos horizontais e verticais.



Transfer learning

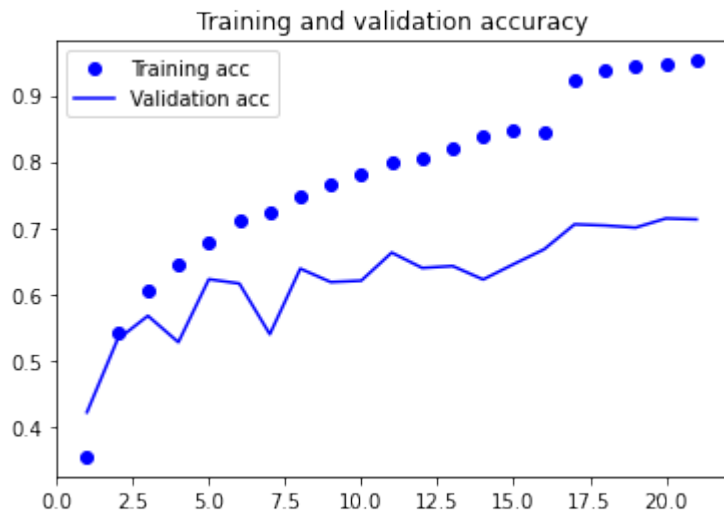
- Transferir o aprendizado de uma rede para outros.
- Muito usado em reconhecimento de imagens.
- Usa os pesos de redes pré treinadas com o dataset imagenet, um dataset de 14 milhões de imagens catalogadas manualmente.
- Permite melhores resultados para reconhecimento de imagens do que uma rede com os parâmetros treinados do zero.



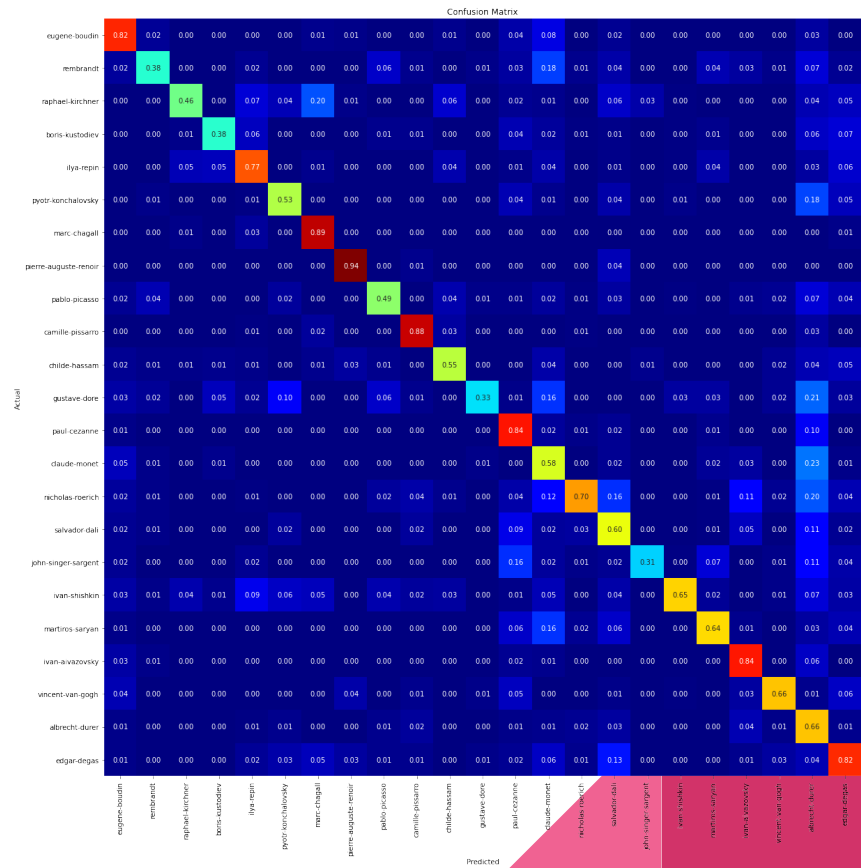
Resultados

Critério	1ª Iteração	2ª Iteração	3ª Iteração	Média	Desvio Padrao
VGG16	79,86%	81,28%	76,45%	79.20%	2.02
Resnet	82.06%	88.17%	93.99%	88.07%	4,877
Inception	98.15%	98.22%	97.83%	98.07%	0.16
Inception Resnet	97.05%	96.62%	97.15%	96.94%	0.22
Xception	91.43%	89.25%	96.87%	92.52%	3.2

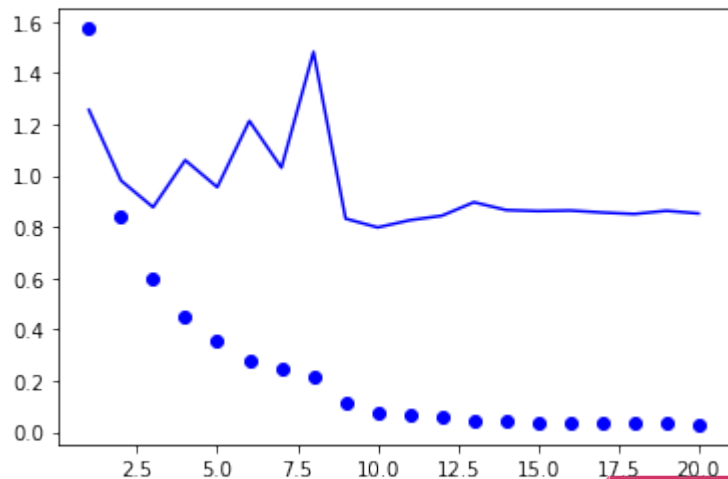
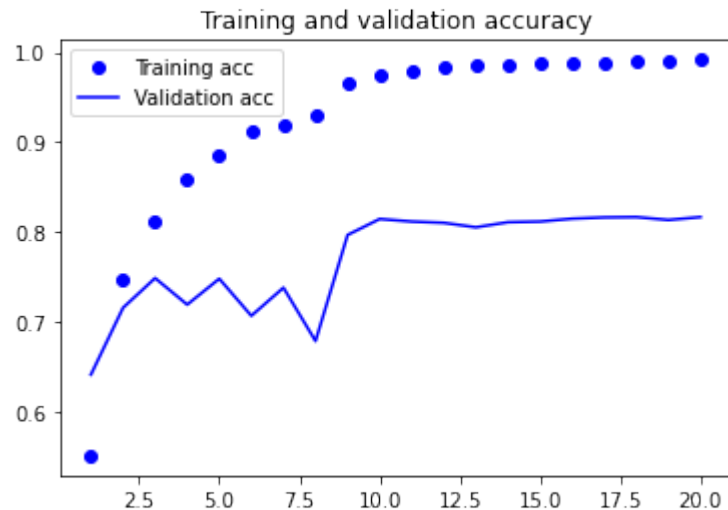
VGG16



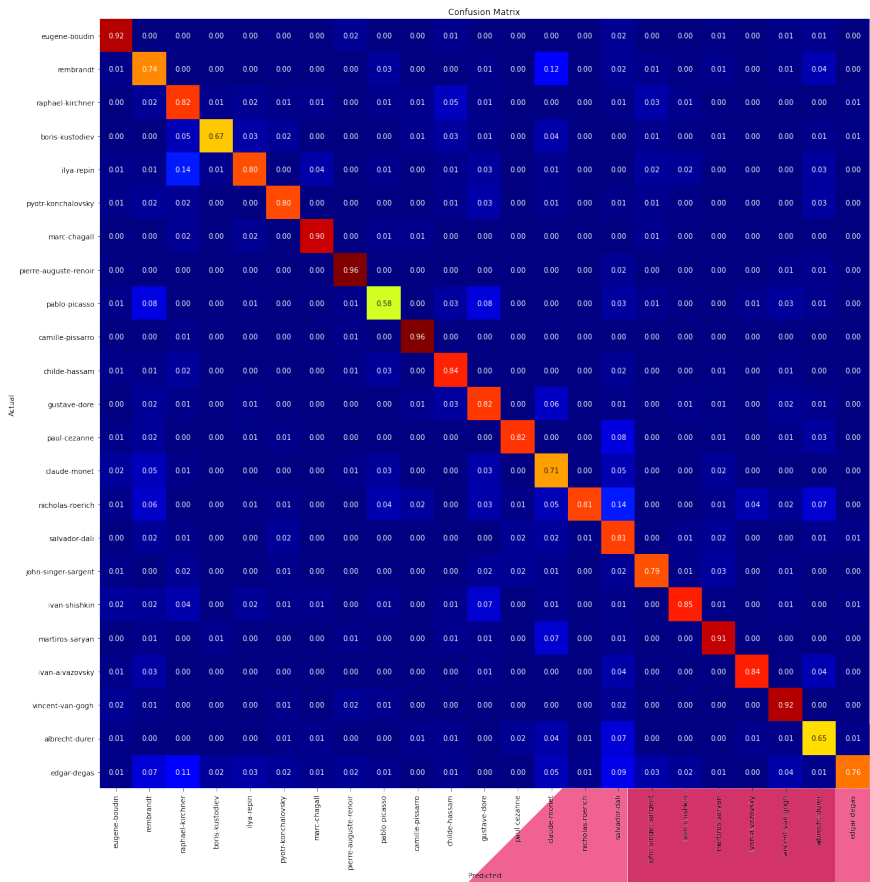
Resultados - VGG16



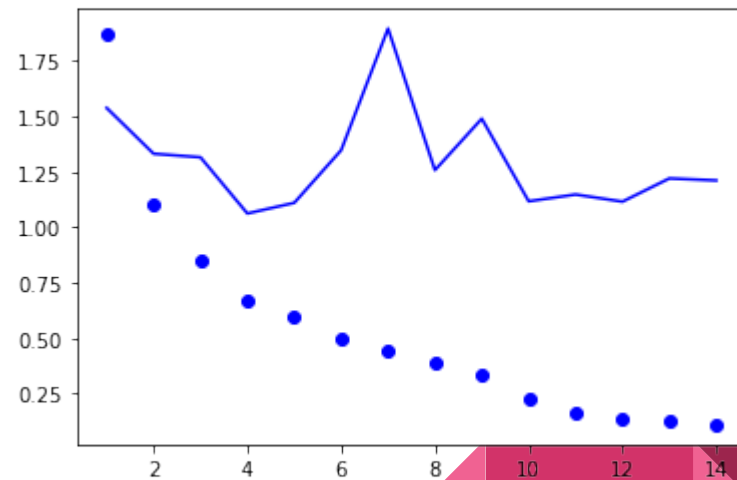
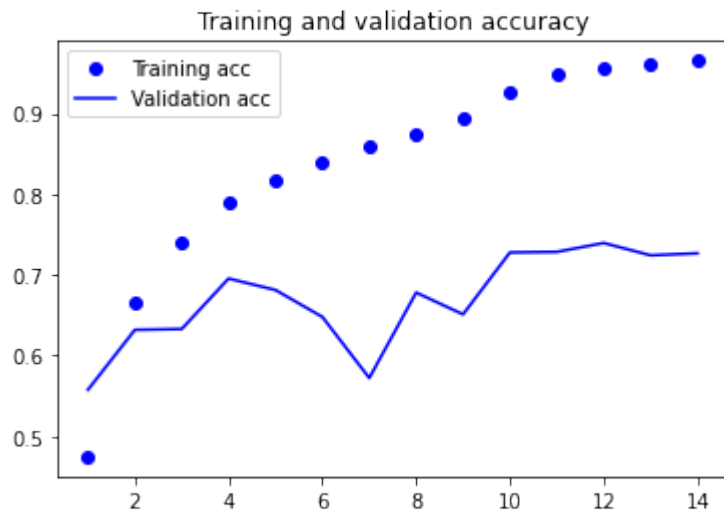
Resultados - Inception



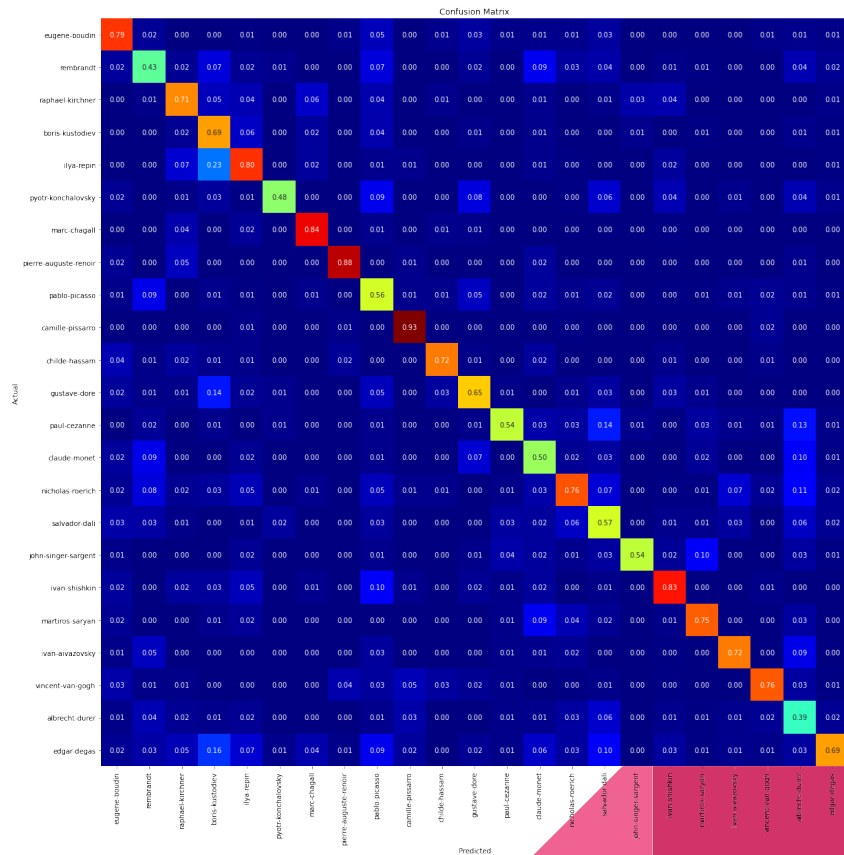
Resultados - Inception



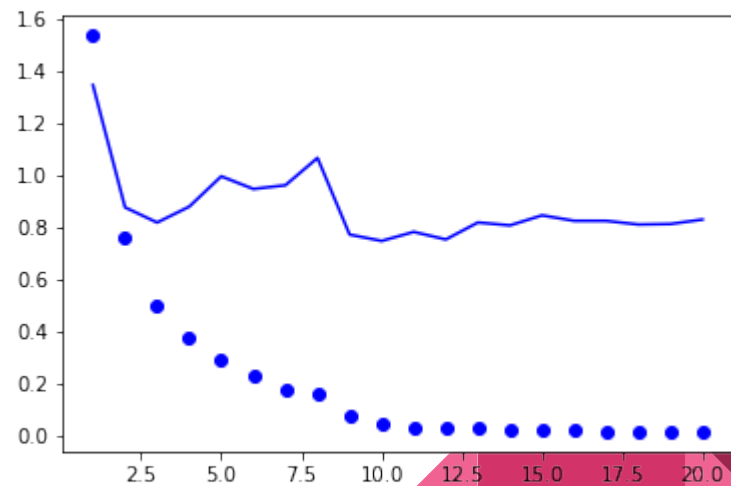
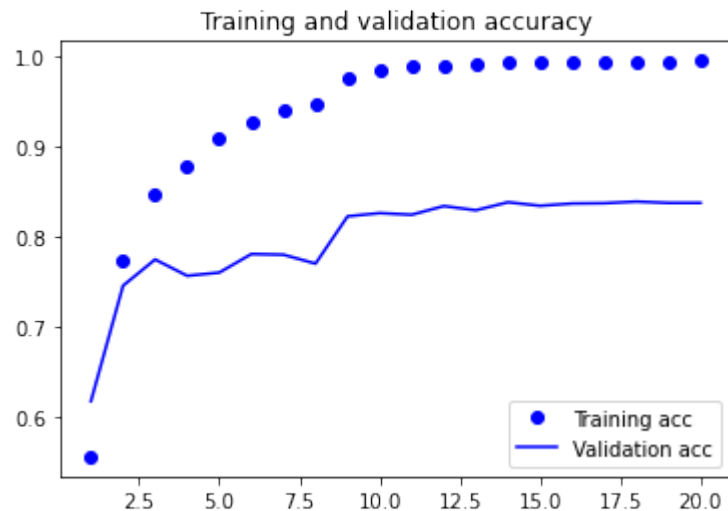
Resultados Resnet



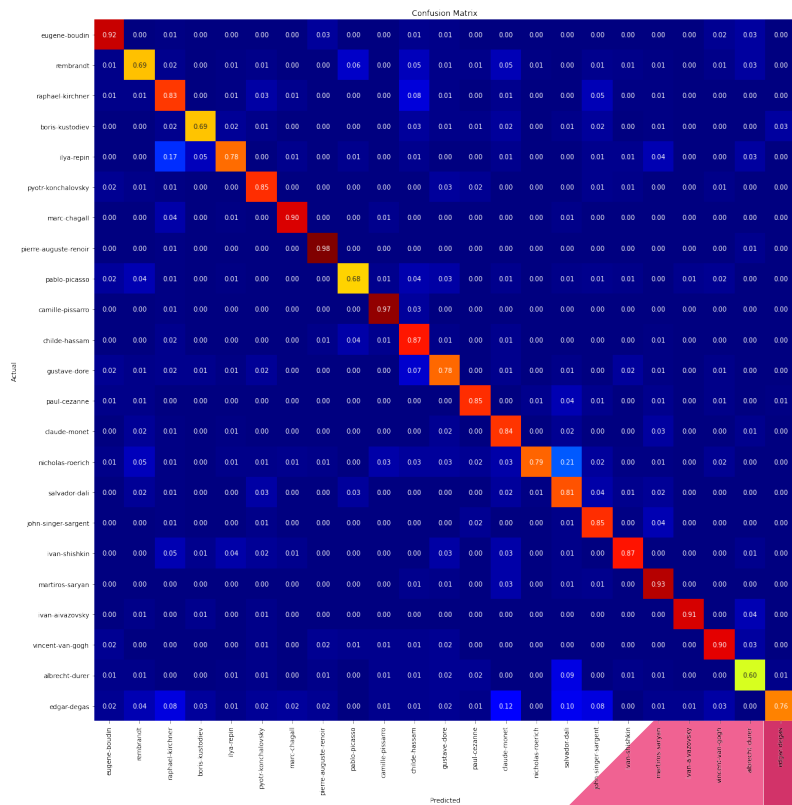
Resultados Resnet



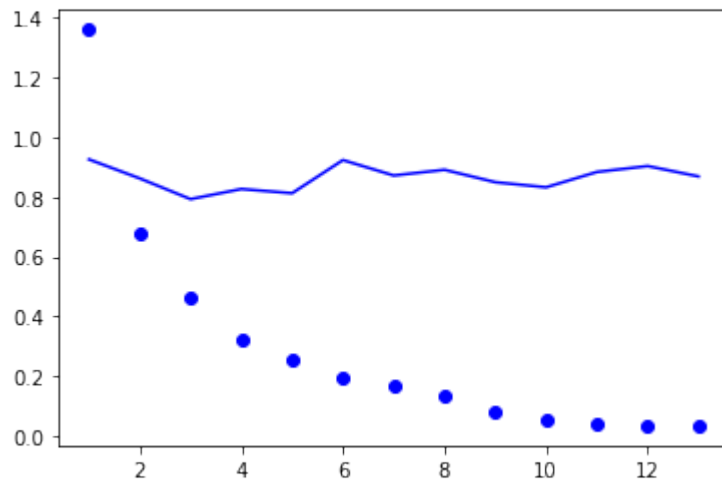
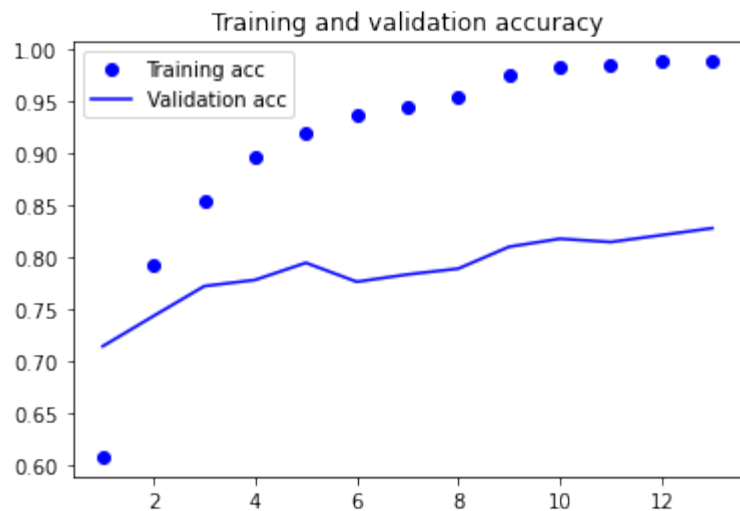
Resultados Resnet Inception



Resultados Resnet Inception



Resultados Xception



Resultados Xception

