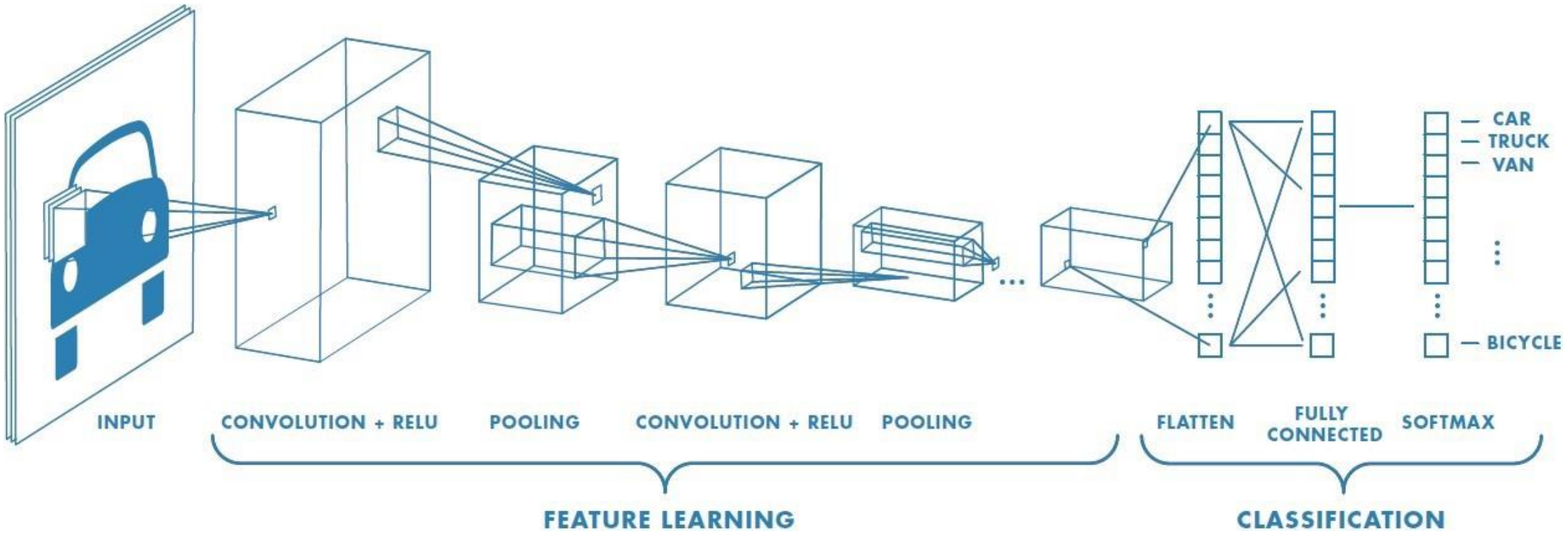
Machine learning

Rede Neural Convolucional



O problema


A_test.jpg
B_test.jpg
C_test.jpg
D_test.jpg
E_test.jpg
F_test.jpg
G_test.jpg
H_test.jpg
I_test.jpg
J_test.jpg
K_test.jpg
L_test.jpg
M_test.jpg
N_test.jpg
nothing_test.jpg
O_test.jpg
P_test.jpg
Q_test.jpg
R_test.jpg
S_test.jpg
space_test.jpg
T_test.jpg
U_test.jpg
V_test.jpg
W_test.jpg
X_test.jpg
Y_test.jpg
Z_test.jpg

Questão 1

`code1.py`

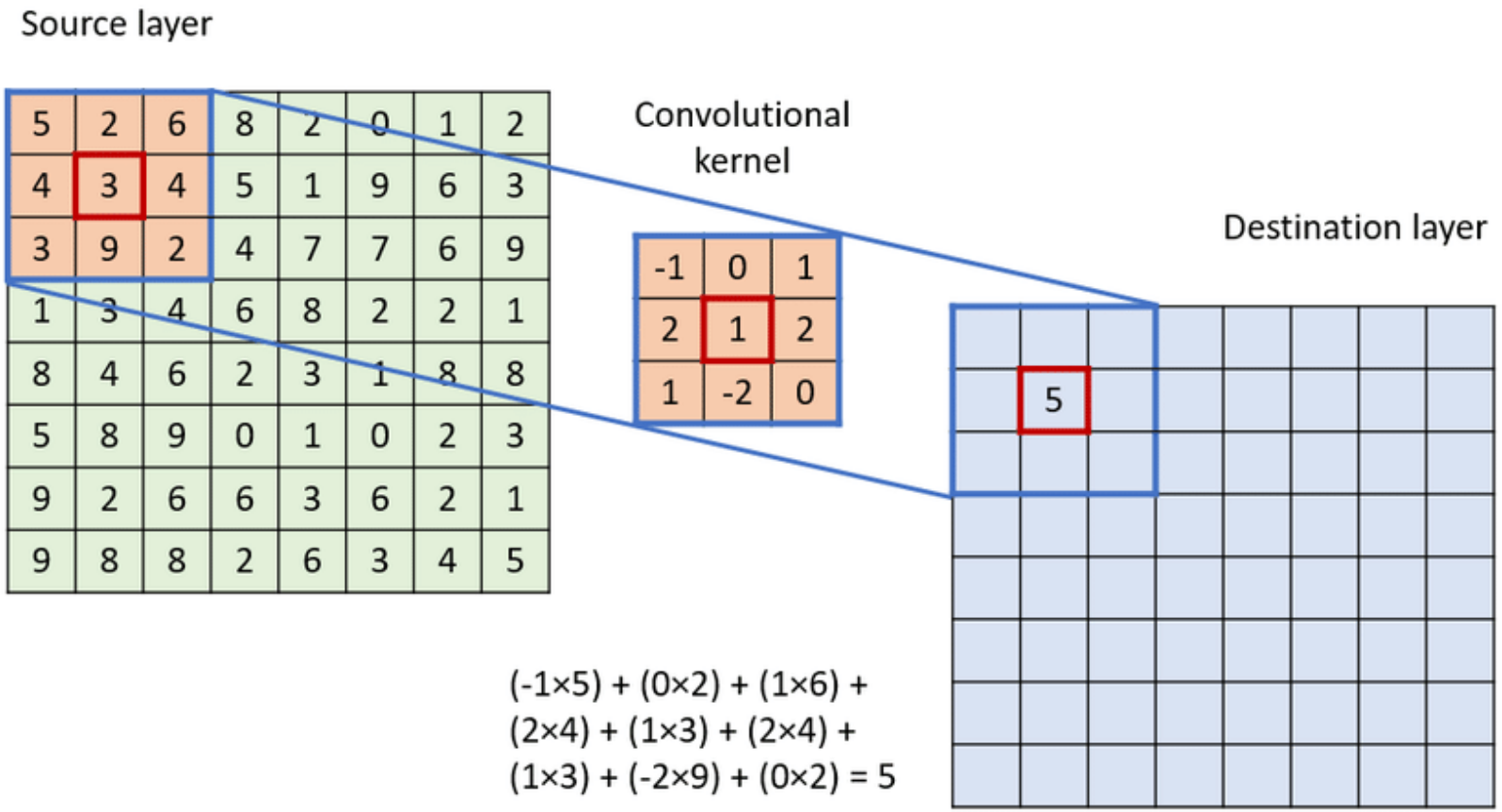
verificação do dataset

Neste primeiro exercício podemos verificar o balanceamento dos dados do dataset.

Instruções:

1. Utilizando o módulo `pandas` crie um dataframe `df` dos dados contidos em `label_train`, usando o método `dataframe()`.
2. utilize a função `hist()` do objeto `df` para calcular o histograma dos atributos.

Convolução


Source layer
5 2 6 8 2 0 1 2
4 3 4 5 1 9 6 3
3 9 2 4 7 7 6 9
1 3 4 6 8 2 2 1
8 4 6 2 3 1 8 8
5 8 9 0 1 0 2 3
9 2 6 6 3 6 2 1
9 8 8 2 6 3 4 5
Convolutional kernel
-1 0 1
2 1 2
1 -2 0
Destination layer
5
(-1×5) + (0×2) + (1×6) +
(2×4) + (1×3) + (2×4) +
(1×3) + (-2×9) + (0×2) = 5

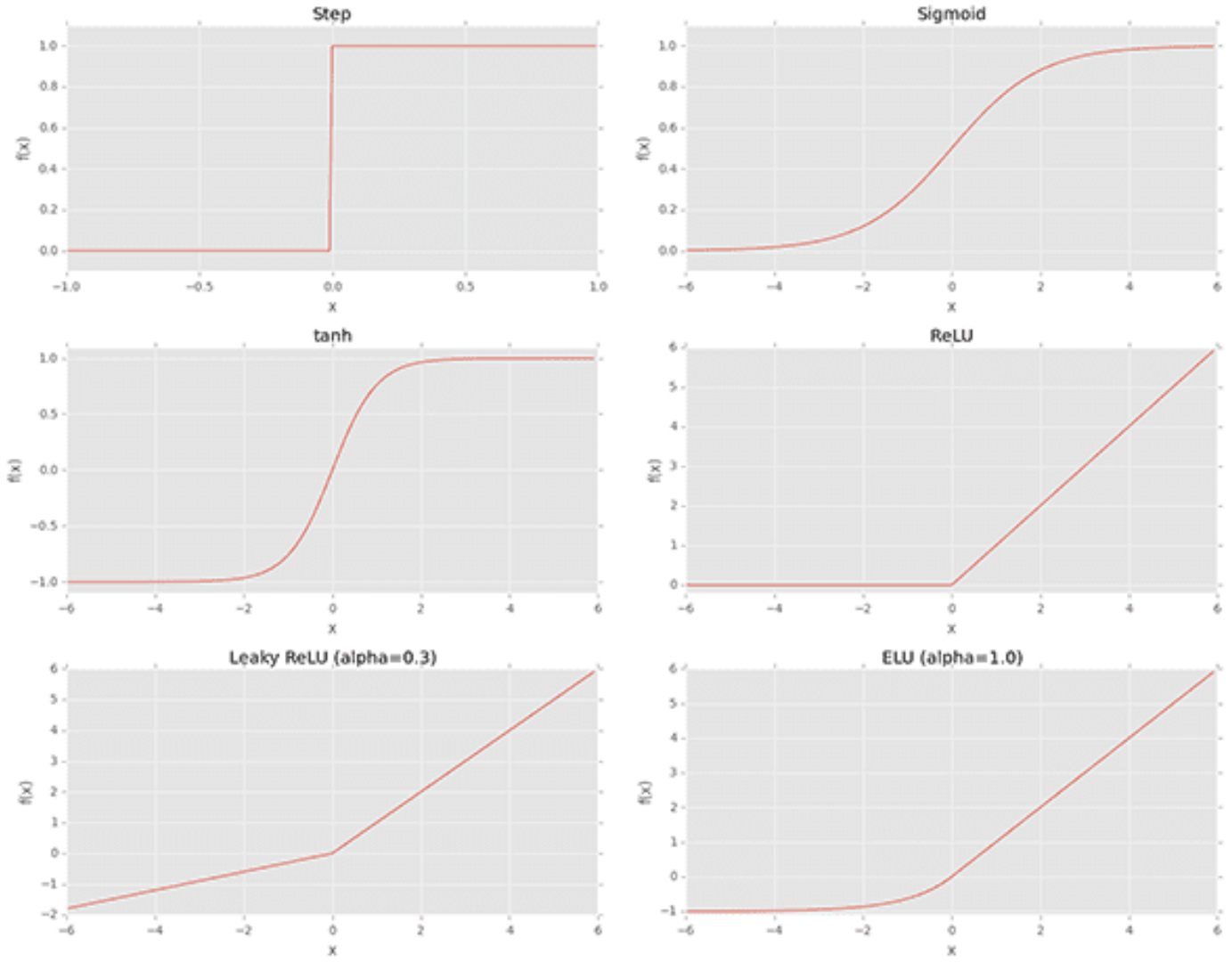
Convolução no Keras

```
Keras Conv2D and Convolutional Layers
 tensorflow.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1),
    padding='valid', data_format=None, dilation_rate=(1, 1),
    activation=None, use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None,
    bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, bias_constraint=None)
```

- **Principais parâmetros.**
- **Filters**: Quantidade de filtros convolucionais que serão utilizados. Utilizar quantidade relativa a potencia de 2 , $filters = 2^n$
- **Kernel:** Tamanho da matriz do filtro. (3x3) , (5x5), (7x7)...
- **Stride**: Indica o passo da convolução
- **Padding:** determina o preenchimento das bordas. Se ‘valid’ a borda é eliminada e a imagem é reduzida. Se ‘same’ a borda é espelhada e a imagem não é reduzida.

Convolução no Keras

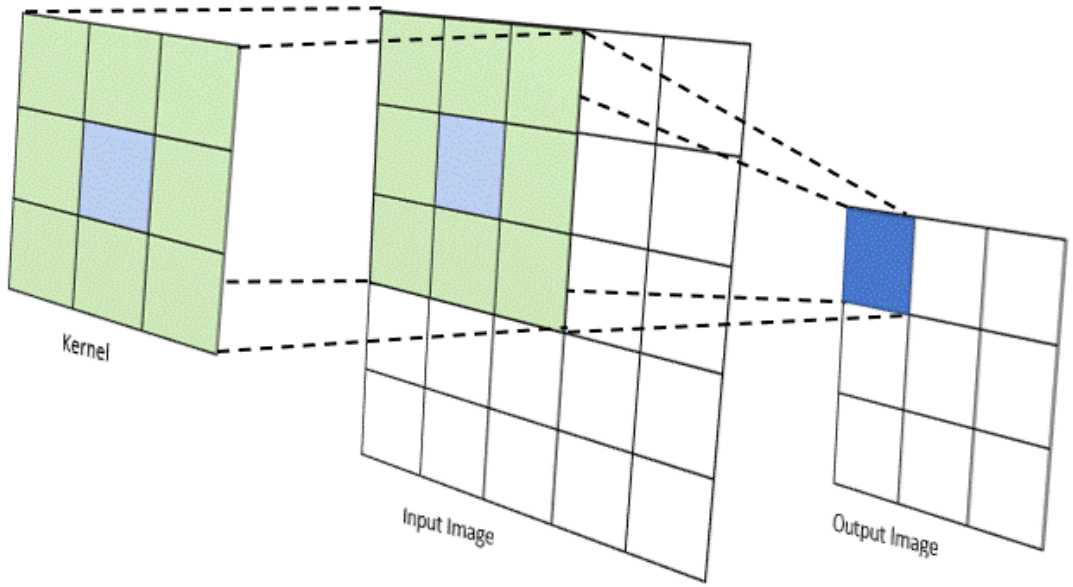
- **Activation:**



Convolução no Keras

- Exemplo

```
model = Sequential()
model.add(Conv2D(32, kernel_size =(5, 5), strides =(1, 1),
                 activation ='relu'))
```

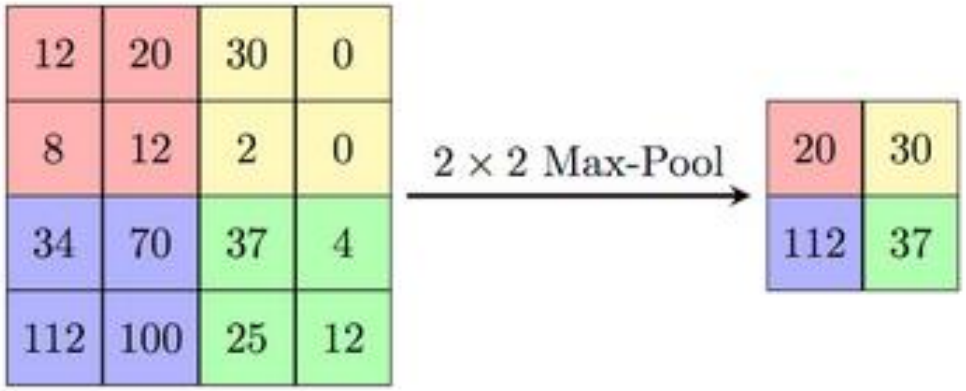


Camada de redução maxpooling

MaxPooling2D layer

`MaxPooling2D` class

```
tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2), strides=None, padding="valid", data_format=None, **kwargs
)
```



Questão 2

`code2.py`

Treinamento de uma rede neural convolucional

O objetivo deste exercicício é realizar um treinamento básico de uma rede neural convolucional

Instruções

1. Inicilize o modelo sequencial do keras com o método `sequential()` do módulo `keras`.
2. Crie a primeira camada convolucional em `model` com `conv2D` de 4 filtros, kernel de tamanho 3x3, função de ativação `relu` e padding definido como `same`.
3. Adicione em `model` uma camada de redução de dimensão `maxpoooling2D`. Defina o tamanho do filtro em 2x2 e defina padding do tipo `same`.
4. transforme os dados em um tensor de atributos unidimensionais. Adicione em `model` uma camada `flatten`.
5. Adicione em `model` a camada de saída, densa com 29 neurônios e função de ativação `softmax`.
6. Realize a compilação do modelo com o método `compile` de `model` defina loss como `sparse_categorical_crossentropy`, utilizo otimizador `adam` e métricas para analizar treinamento `sparse_categorical_accuracy`.
7. Implemente o treinamento do modelo com o método `fit` atribua `data_train` como dados de treinamento, `label_train` como rótulos dos dados a serem treinados, os dados de validação são `data_val` e `label_val`.Defina 5 épocas.

Questão 3

`code3.py`

Salvando um modelo durante o treinamento

Neste exercicio você vai realizar o treinamento de um modelo e salvar o melhor modelo com base na epóca que obteve o melhor acerto.

Instruções

1. Inicialize o modelo sequencial do keras com o método `sequential()` do módulo `keras`.
2. Crie a primeira camada convolucional em `model` com `conv2D` de 16 filtros, kernel de tamanho 5x5, função de ativação `relu` e padding definido como `same`.
3. Adicione em `model` uma camada de redução de dimensão `maxpoooling2D`. Defina o tamanho do filtro em 2x2 e defina padding do tipo `same`.
4. transforme os dados em um tensor de atributos unidimensionais. Adicione em `model` uma camada `flatten`.
5. Adicione em `model` a camada de saída, densa com 29 neurônios e função de ativação `softmax`.
6. Realize a compilação do modelo com o método `compile` de `model` defina loss como `sparse_categorical_crossentropy`, utilize otimizador `RMSprop` e métricas para analizar treinamento `sparse_categorical_accuracy`.
7. atribua um nome para seu modelo de sua preferencia
8. Utilize o método `ModelCheckpoint` para salvar o modelo após verificar que este não reduziu a mais o valor do parâmetro `val_loss`. Essa verificação é realizado após 5 épocas que é definido em `patience`. Atribuindo `verbose` com valor 1 será exibido no terminal quando for salvo algum modelo. Defina `mode` igual a `min` para informar que deseja salvar o modelo com a moneor perda possível.
9. Utilize o método `EarlyStopping` para interromper o treinamento quando não for observada mais evolução no aprendizado. O parâmetros `val_loss` é o parâmetro que deve ser observado , a quatidade de épocas é 5 verbose igual 1 e mode `min`.
10. Utilizr o método `ReduceLROnPlateau` para reduzir a taxa de aprendizagem quando o modelo não aprender mais durante 5 épocas. O valor de melhoria que deve ser é observado é definido em `factor` que deve ser 0.05.
11. Implemente o treinamento do modelo com o método `fit` de `model`. Atribua `data_train` como dados de treinamento, `label_train` como rótulos dos dados a serem treinados, os dados de validação são `data_val` e `label_val` .Defina 50 épocas. Defina em `callbacks` os paramentros de parada de treinamento, salvamento do modelo e redução da taxa de aprendizagem.

Questão 4

`code4.py`

Utilização do tensorboard

Neste exercício você irá utilizar o tensorboard para monitorar e registrar o comportamento da rede neural durante o processo de aprendizagem.

Instruções

1. Inicialize o modelo sequencial.
2. Adicione a primeira camada convolucional com quantidade de fitros da sua escolha (potência de 2), bem como o tamanho do filtro.
3. adicione a camada maxpooling2D com tamanho 2x2.
4. transforme os dados em 1 dimensão
5. Adicione a camada de saída densa com 29 neuronios e ativação softmax
6. Compile o modelo.
7. Defina um nome para o modelo
8. Utilize o método `TensorBoard` e configure o caminho para salvar o log com o nome do modelo que está em `model_file`.
9. Defina os critérios de parada do treinamento, salvamento do modelo e redução da taxa de aprendizagem
10. realize o treinamento, lembre-se de incluir `tensorBoard` no `callbacks`.

Questão 5

`code5.py`

Grid search simplificado para deteminar quantidade de filtros de blocos convolucionais

Vamos ver neste exercicio alguns atributos que podem ser extraídos da imagem, como média, variância e dados do histograma.

Instruções

1. Defina em `conv_node` a quantidade possivel de filtros convolucionais. Considere 8,16,32,64.
2. Em output determine a quantidade de neurônios da ultima camada de saída.
3. Em first_node determine 8.
4. Inicialize o modelo sequencial.
5. crie a primeira camada convolucional com input_shape dos dados da imagem de entrada e quantidade de filtros definida em first_node. Ativação relu, kernel 3x3 e padding same.
6. Adicione as camadas maxpooling e dropout.
7. Na iteração para criar blocos adicionais de caamdas convolucionais crie a camada Conv2D de modo que a quantodade de filtros é iterativa atribuido a variável node para iterar na parametrização da quantidade de filtros. Crie posteriormente a camada maxpooling2D
8. Adicione a camada Flatten
9. Adicione a camada de saída densa com neuronios definidos em output. Ativação softmax.
10. Inclua model_name no argumento da função callbacks
11. Inclua model_name no argumento da função save_sumary
12. execute o treinamento considerando 5 épocas.

Questão 6

`code6.py`

Visualização dos filtros e imagem resultado das camadas convolucionais

Vamos ver neste exercício alguns atributos que podem ser extraídos da imagem, como média, variância e dados do histograma.

Instruções

1. Utilize o método `load_model` do keras para carregar um modelo salvo treinado na questão anterior
2. Acesse a primeira camada convolucional com o método `layers`.
3. Após visualizados os filtros, utilize novamente o método layers para inicializar o modelo de forma que a saída deste seja a ultima camada convolucional. Veja o sumário do seu modelo e atribua o indice da ultima linha que existe uma camada convolucional.
4. faça uma predição em data_test e armazene em features.
5. armazene em features_maps a decima predição
6. mostre a decima imagem original
7. moestre a imagem da predicão passando feature_maps para show_features