Projeto Interdisciplinar: CIFAR-10

Daniel Brai Gonzales Marcos	CP301335
Giovanna Nascimento Antonieti	CP3013383

Agenda

1. Introdução
   Base de dados

2. Metodologia
   Baseline e melhorias

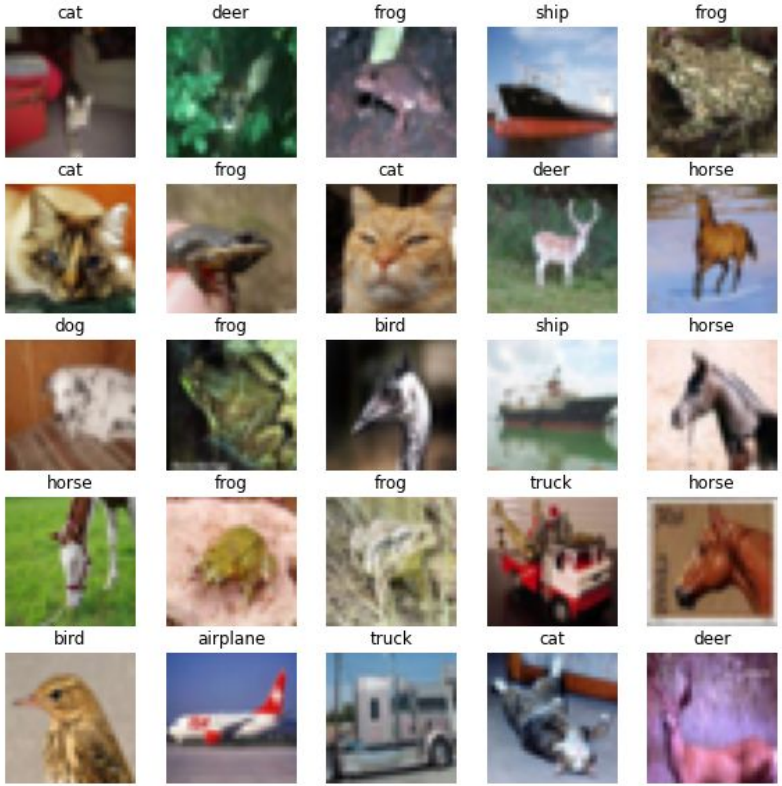
3. Resultados
   Onde os modelos analisados chegaram

4. Conclusões
   Onde os modelos analisados chegaram

Base de dados

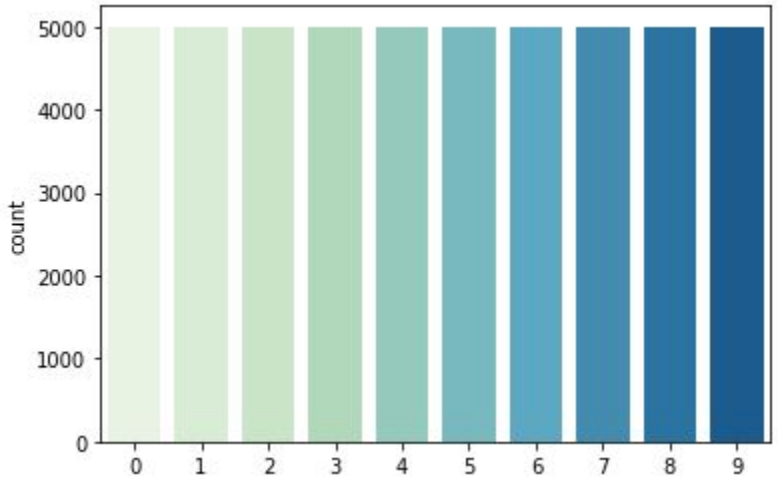
- 60.000 imagens coloridas;
  - 50.000 -> treino
  - 10.000 -> teste
- 32x32;
- 10 classes
  - sem sobreposição entre classes.
  - avião, automóvel, pássaro, veado, gato, cachorro, sapo, cavalo, navio e caminhão.

Base de dados

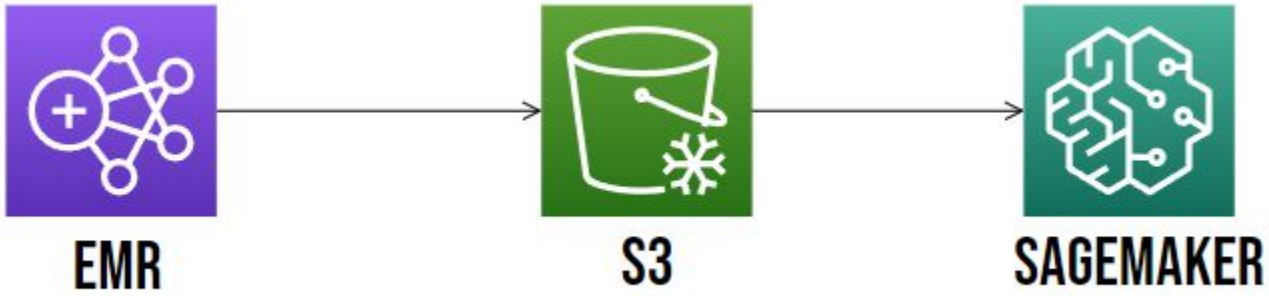


Base de dados

Distribuição dos dados nas classes:



Pipeline AWS



Redes neurais

- Redes convolucionais;

- VGG [1] -> ILSVRC 2014;
  - 2 camadas convolucionais com filtros de 3x3;
  - Camada de *max pooling* (condensando a saída da camada anterior)

---

[1] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Redes neurais

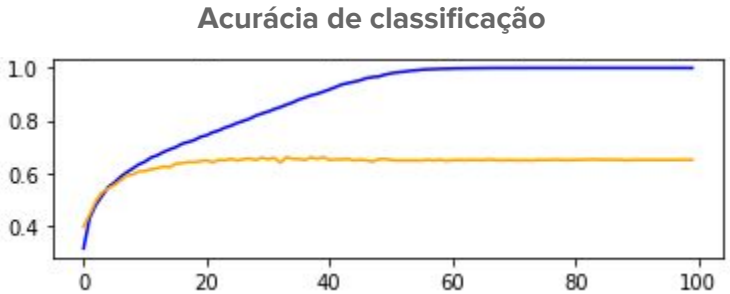
- Modelos:
  - 1 bloco VGG;
  - 2 blocos VGG;
  - 3 blocos de VGG.

- SGD
  - *momentum*=0.9
  - *learning rate*=0.0001

[1] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

1 bloco VGG

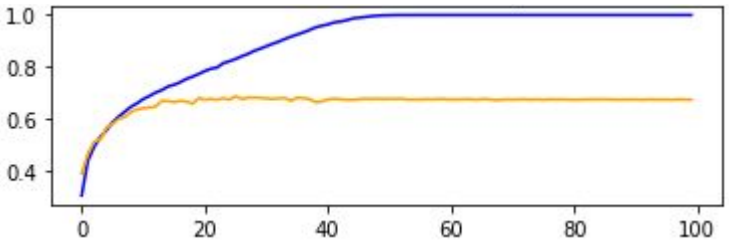
```
model_vgg1 = Sequential()
model_vgg1.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model_vgg1.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg1.add(MaxPooling2D((2, 2)))
model_vgg1.add(Flatten())
model_vgg1.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model_vgg1.add(Dense(10, activation='softmax'))
opt = SGD(lr=0.001, momentum=0.9)
model_vgg1.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```



2 bloco VGG

```
model_vgg2 = Sequential()
model_vgg2.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model_vgg2.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg2.add(MaxPooling2D((2, 2)))
model_vgg2.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg2.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg2.add(MaxPooling2D((2, 2)))
model_vgg2.add(Flatten())
model_vgg2.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model_vgg2.add(Dense(10, activation='softmax'))
model_vgg2.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

Acurácia de classificação



3 bloco VGG

```
model_vgg3 = Sequential()
model_vgg3.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model_vgg3.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg3.add(MaxPooling2D((2, 2)))
model_vgg3.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg3.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg3.add(MaxPooling2D((2, 2)))
model_vgg3.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg3.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_vgg3.add(MaxPooling2D((2, 2)))
model_vgg3.add(Flatten())
model_vgg3.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model_vgg3.add(Dense(10, activation='softmax'))
model_vgg3.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

Discussão

- Aumento do desempenho conforme aumento da profundidade do modelo;

- O modelo foi capaz de aprender o conjunto de dados de treinamento
  - 40 épocas;

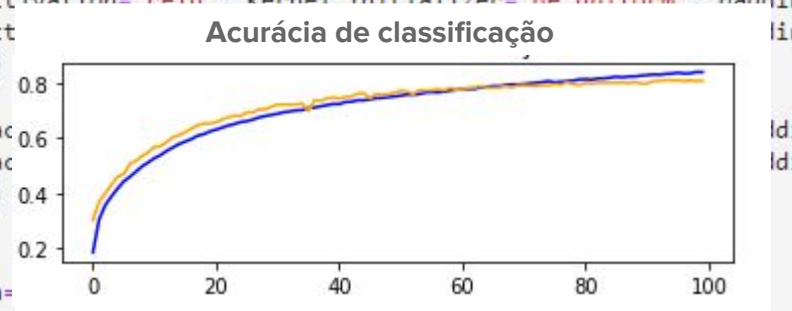
- Todos os três modelos mostraram o mesmo padrão de overfitting
  - 15 - 20 épocas.

Melhorias: Dropout

```
model_dropout = Sequential()
model_dropout.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model_dropout.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(MaxPooling2D((2, 2)))
model_dropout.add(Dropout(0.2))
model_dropout.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(MaxPooling2D((2, 2)))
model_dropout.add(Dropout(0.2))
model_dropout.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(MaxPooling2D((2, 2)))
model_dropout.add(Dropout(0.2))
model_dropout.add(Flatten())
model_dropout.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model_dropout.add(Dropout(0.2))
model_dropout.add(Dense(10, activation='softmax'))
model_dropout.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

Melhorias: Dropout

```
model_dropout = Sequential()
model_dropout.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model_dropout.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(MaxPooling2D((2, 2)))
model_dropout.add(Dropout(0.2))
model_dropout.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_dropout.add(Conv2D(64, (3, 3), act                                                  ding='same'))
model_dropout.add(MaxPooling2D((2, 2)))
model_dropout.add(Dropout(0.2))
model_dropout.add(Conv2D(128, (3, 3), ac                                                  dding='same'))
model_dropout.add(Conv2D(128, (3, 3), ac                                                  dding='same'))
model_dropout.add(MaxPooling2D((2, 2)))
model_dropout.add(Dropout(0.2))
model_dropout.add(Flatten())
model_dropout.add(Dense(128, activation=
model_dropout.add(Dropout(0.2))
model_dropout.add(Dense(10, activation='softmax'))
model_dropout.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

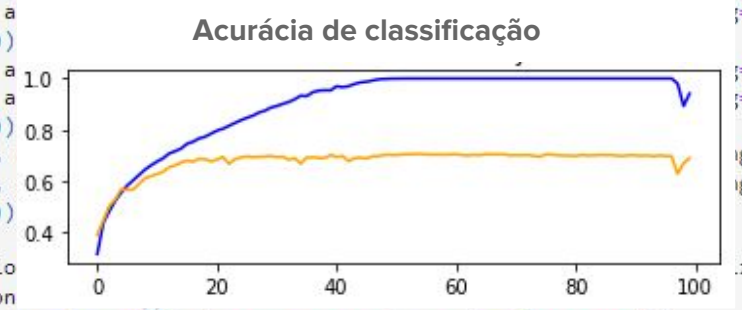


Melhorias: Regularização

```
model_regularization = Sequential()
model_regularization.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001), input_
model_regularization.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model_regularization.add(MaxPooling2D((2, 2)))
model_regularization.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model_regularization.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model_regularization.add(MaxPooling2D((2, 2)))
model_regularization.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model_regularization.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model_regularization.add(MaxPooling2D((2, 2)))
model_regularization.add(Flatten())
model_regularization.add(Dense(128, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
model_regularization.add(Dense(10, activation='softmax'))
model_regularization.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

Melhorias: Regularização

```
model_regularization = Sequential()
model_regularization.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001), input_
model_regularization.add(Conv2D(32, (3, 3), a                                                  ='same', kernel_regularizer=l2(0.001)))
model_regularization.add(MaxPooling2D((2, 2))
model_regularization.add(Conv2D(64, (3, 3), a                                                  ='same', kernel_regularizer=l2(0.001)))
model_regularization.add(Conv2D(64, (3, 3), a                                                  ='same', kernel_regularizer=l2(0.001)))
model_regularization.add(MaxPooling2D((2, 2))
model_regularization.add(Conv2D(128, (3, 3),                                                   g='same', kernel_regularizer=l2(0.001)))
model_regularization.add(Conv2D(128, (3, 3),                                                   g='same', kernel_regularizer=l2(0.001)))
model_regularization.add(MaxPooling2D((2, 2))
model_regularization.add(Flatten())
model_regularization.add(Dense(128, activatio                                                  zer=l2(0.001)))
model_regularization.add(Dense(10, activation
model_regularization.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

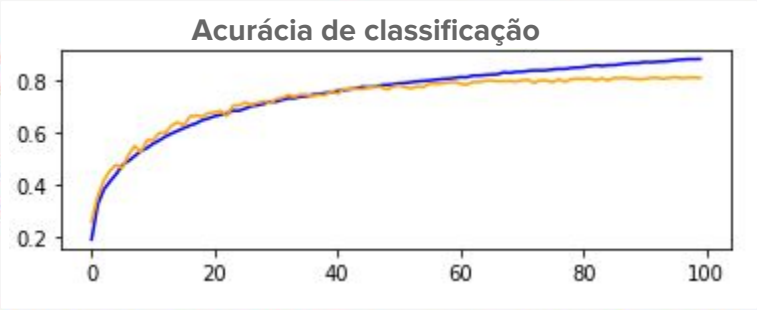


Melhorias: Dropout + Regularização

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001), input_shape=(32, 32,
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

Melhorias: Dropout + Regularização

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001), input_shape=(32, 32,
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(0.001)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='rel                                                    _regularizer=l2(0.001)))
model.add(Conv2D(64, (3, 3), activation='rel                                                    _regularizer=l2(0.001)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='re                                                    l_regularizer=l2(0.001)))
model.add(Conv2D(128, (3, 3), activation='re                                                    l_regularizer=l2(0.001)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```



Avaliação no conjunto de teste

- Baseline: 3 blocos da VGG - 69%
- Dropout - 79%
- Dropout + Regularização - 79%

Próximos passos

- Explorar outras taxas de dropout, bem como diferentes posicionamentos dessas camadas na arquitetura do modelo;

- Tentar melhorar o efeito da regularização talvez usando uma ponderação maior, como 0,01 ou mesmo 0,1.

- *Transfer learning* -> VGG16