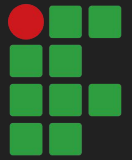
Bacharelado em Ciência da Computação
Disciplina: Inteligência Artificial



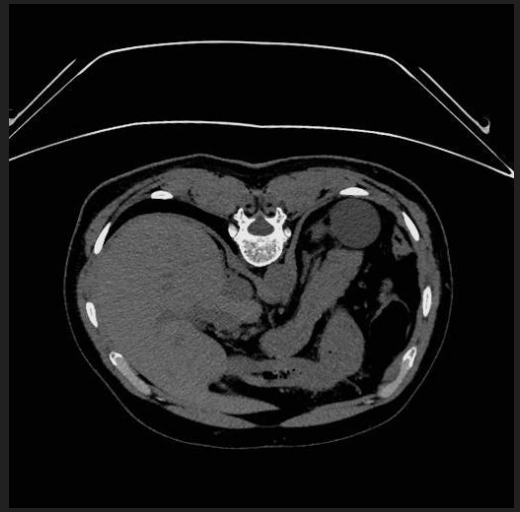
TRABALHO PRÁTICO SOBRE REDES NEURAIS CONVOLUCIONAIS

Discentes: João Gabriel Garcia de Sousa, Mateus Cesar Marques, Yuri Neres Macedo
Docente: Hugo Resende

Datasets

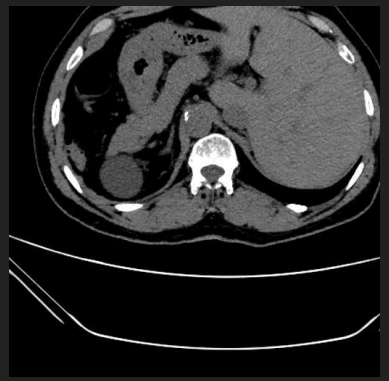
Car make, model, and generation

Multicancer



Multicâncer

kidney Cyst



kidney Normal

kidney Stone

kidney Tumor.

Data Augmentation

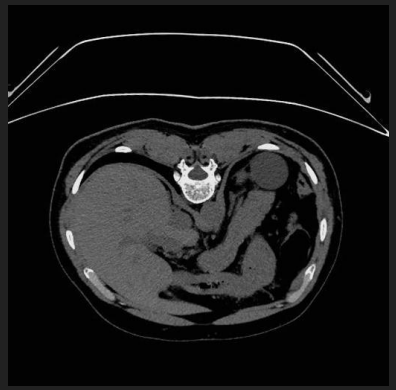
	antes	depois
kidney cyst	train = 4000	train = 2000
	val = 556	val = 500
	test = 557	test = 500
kidney tumor	train = 4000	train = 2000
	val = 342	val = 500
	test = 343	test = 500
kidney stone	train = 4000	train = 2000
	val = 206	val = 500
	test = 208	test = 500
kidney normal	train = 4000	train = 2000
	val = 761	val = 500
	test = 763	test = 500

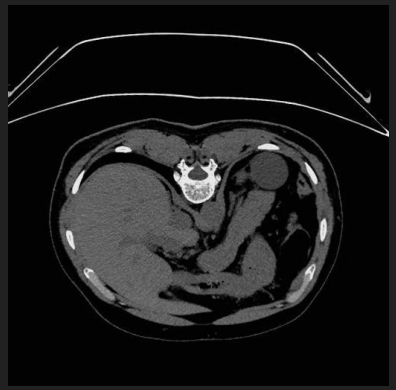
Parâmetros

- chance de 50% de inverter horizontalmente.
- chance de 50% de inverter verticalmente.
- rotação das imagens entre -25 e 25 graus.
- alterar o brilho entre 0.8 e 1.2.
- desfoque gaussiano entre 0 e 1.

Pré-Processamento das imagens

Durante a criação dos train, test e val generator as imagens que são RGB são salvas em escala de cinza nos generator.

Escala de cinza 

RGB 

Pré-Processamento das imagens

```
general_datagen = ImageDataGenerator(
    rescale=1./255,
    preprocessing_function=preprocess_image
)

train_generator = general_datagen.flow_from_directory(
    train_directory,
    target_size=(224, 224),
    batch_size=32,
    color_mode='grayscale'
)

valid_generator = general_datagen.flow_from_directory(
    val_directory,
    target_size=(224, 224),
    batch_size=32,
    color_mode='grayscale'
)
```

```
test_generator = general_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    color_mode='grayscale',
    shuffle=False
)
```

```
Found 8005 images belonging to 4 classes.
Found 2000 images belonging to 4 classes.
Found 2000 images belonging to 4 classes.
```

```
Train groups: 251
Validation groups: 63
Test groups: 63
```

Arquitetura

```python
def conv_layer(inputs, filters, kernel_size=3, padding="valid"):
    x = layers.Conv2D(filters = filters, kernel_size = kernel_size, padding = padding, use_bias = False)(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)
    return x

def pooling_layer(inputs, pool_size = 2, dropout_rate=0.5):
    x = layers.MaxPooling2D(pool_size = pool_size)(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Dropout(dropout_rate)(x)
    return x

def dense_layer(inputs, out, dropout_rate = 0.5):
    x = layers.Dense(out)(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)
    x = layers.Dropout(dropout_rate)(x)
    return x
```

Arquitetura

```
keras.backend.clear_session()
inputs = keras.Input(shape = (224, 224, 1))
x = conv_layer(inputs, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")

x = layers.Flatten()(x)
x = dense_layer(x, 96)

outputs = layers.Dense(category_count, activation = "softmax")(x)
base_model = keras.Model(inputs, outputs)
base_model.summary()
```

Treinamento

```
base_model.compile(optimizer =keras.optimizers.Adam(learning_rate=0.001),
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])

history = base_model.fit(
    train_generator,
    steps_per_epoch = train_groups,
    epochs = 20,
    validation_data = valid_generator,
    validation_steps = valid_groups,
    verbose = 1,
    callbacks=[keras.callbacks.EarlyStopping(monitor='val_accuracy', patience = 10, restore_best_weights = True),
              keras.callbacks.ReduceLROnPlateau(monitor = 'val_loss', factor = 0.7, patience = 2, verbose = 1),
    keras.callbacks.ModelCheckpoint(
            filepath = "/content/drive/MyDrive/Datasets/rim/model_rim.h5",
            save_best_only = True,
            monitor = "val_loss")
    ])
```

```
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.00024009999469853935.
251/251 [==============================] - 107s 427ms/step - loss: 0.0754 - accuracy: 0.9775 - val_loss: 0.4494 - val_accuracy: 0.8885 - lr: 3.4300e-04
Epoch 20/20
251/251 [==============================] - 107s 425ms/step - loss: 0.0697 - accuracy: 0.9794 - val_loss: 0.3115 - val_accuracy: 0.8925 - lr: 2.4010e-04
```

Métricas obtidas

```
model = keras.models.load_model("/content/drive/MyDrive/rim/model_rim2.h5")
test_results = model.evaluate(test_generator)
loss, accuracy = test_results
print(f'Perda (Loss): {loss}')
print(f'Acurácia: {accuracy}')
```

```
63/63 [==============================] - 331s 5s/step - loss: 0.2462 - accuracy: 0.9155
Perda (Loss): 0.2461744099855423
Acurácia: 0.9154999852180481
```

```
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_generator.classes
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(true_classes, predicted_classes)
print(f'Precisão: {accuracy}')
```

```
63/63 [==============================] - 10s 157ms/step
Precisão: 0.9155
```

```
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_generator.classes
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(true_classes, predicted_classes)
print(f'Precisão: {accuracy}')
```

```
63/63 [==============================] - 10s 157ms/step
Precisão: 0.9155
```

Métricas obtidas

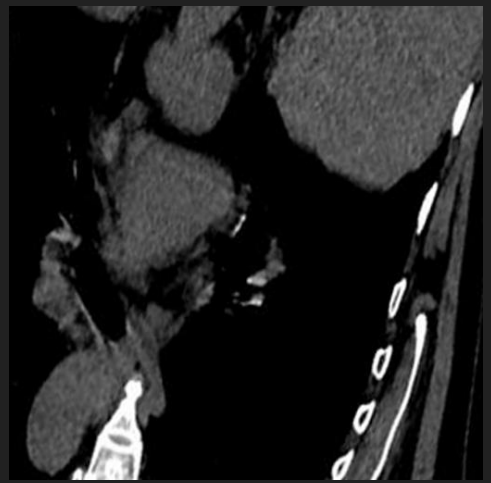
	kidney cyst	kidney normal	kidney stone	kidney tumor
kidney cyst	480	0	20	0
kidney normal	2	465	17	16
kidney stone	23	14	423	40
kidney tumor	14	21	2	463

```
from sklearn.metrics import confusion_matrix
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_generator.classes
confusion = confusion_matrix(true_classes, predicted_classes)
print('Matriz de Confusão:')
print(confusion)
```

```
63/63 [==============================] - 15s 245ms/step
Matriz de Confusão:
[[480   0  20   0]
 [  2 465  17  16]
 [ 23  14 423  40]
 [ 14  21   2 463]]
```

Justificativa

- O dataset possui imagens de tomografia não só do rim mas algumas de uma grande parte do corpo.
- O data Augmentation foi usado apenas no teste e validação então o treino pode não ter sido calibrado para as imagens modificadas.



Car make, model, and generation

audi a7

bmw série 7

dodge charger

porsche 911

Data augmentation

	antes	depois
audi a7	635	train = 1000
		val = 250
		test = 250
bmw série 7	426	train = 1000
		val = 250
		test = 250
dodge charger	237	train = 1000
		val = 250
		test = 250
porsche 911	438	train = 1000
		val = 250
		test = 250

Parâmetros

- chance de 50% de inverter horizontalmente.
- chance de 50% de inverter verticalmente.
- rotação das imagens entre -25 e 25 graus.
- alterar o brilho entre 0.8 e 1.2.
- desfoque gaussiano entre 0 e 1.

Pré-Processamento das imagens

```
augmented_gen = ImageDataGenerator(
    rescale=1./255)

general_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = general_datagen.flow_from_directory(
    train_directory,
    target_size = (250, 250),
    batch_size = 16
)
valid_generator = general_datagen.flow_from_directory(
    val_directory,
    target_size = (250, 250),
    batch_size = 16
)
```

```
test_generator = general_datagen.flow_from_directory(
    test_directory,
    target_size = (250, 250),
    batch_size = 16,
    shuffle=False
)
```

```
Found 4000 images belonging to 4 classes.
Found 1000 images belonging to 4 classes.
Found 1000 images belonging to 4 classes.
```

```
Train groups: 250
Validation groups: 63
Test groups: 63
```

Arquitetura

```
keras.backend.clear_session()
inputs = keras.Input(shape = (250, 250, 3))
x = conv_layer(inputs, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")
x = conv_layer(x, 64)
x = pooling_layer(x)
x = conv_layer(x, 64, padding = "same")

x = layers.Flatten()(x)
x = dense_layer(x, 128)

outputs = layers.Dense(category_count, activation = "softmax")(x)
base_model = keras.Model(inputs, outputs)
base_model.summary()
```

Treinamento

```
base_model.compile(optimizer =keras.optimizers.Adam(learning_rate=0.001),
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])
#fit model
history = base_model.fit(
    train_generator,
    steps_per_epoch = train_groups,
    epochs = 20, # adding more epochs will increase the acc like 1% or 2%
    validation_data = valid_generator,
    validation_steps = valid_groups,
    verbose = 1,
    callbacks=[keras.callbacks.EarlyStopping(monitor='val_accuracy', patience = 7, restore_best_weights = True),
              keras.callbacks.ReduceLROnPlateau(monitor = 'val_loss', factor = 0.7, patience = 7, verbose = 1),
    keras.callbacks.ModelCheckpoint(
            filepath = "/content/drive/MyDrive/Datasets/carro pronto/model_carro.h5",
            save_best_only = True,
            monitor = "val_loss")
    ])
```

```
Epoch 18: ReduceLROnPlateau reducing learning rate to 0.0007000000332482159.
250/250 [==============================] - 69s 277ms/step - loss: 0.0947 - accuracy: 0.9682 - val_loss: 0.2161 - val_accuracy: 0.9270 - lr: 0.0010
```

Métricas obtidas

```
model = keras.models.load_model("/content/drive/MyDrive/Datasets/carro pronto/model_carro.h5")
test_results = model.evaluate(test_generator)
loss, accuracy = test_results
print(f'Perda (Loss): {loss}')
print(f'Acurácia: {accuracy}')
```

```
63/63 [==============================] - 6s 93ms/step - loss: 0.2034 - accuracy: 0.9440
Perda (Loss): 0.20335140824317932
Acurácia: 0.9440000057220459
```

```
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_generator.classes
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(true_classes, predicted_classes)
print(f'Precisão: {accuracy}')
```

```
63/63 [==============================] - 6s 97ms/step
Precisão: 0.944
```

```
from sklearn.metrics import f1_score
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_generator.classes
f1 = f1_score(true_classes, predicted_classes, average='weighted')
print(f'F1-Score: {f1}')
```

```
63/63 [==============================] - 6s 96ms/step
F1-Score: 0.9443621051499442
```

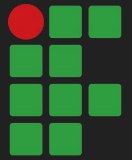
Métricas obtidas

	audi a7	bmw série 7	dodge charger	porsche 911
audi a7	246	0	1	3
bmw série 7	12	237	0	1
dodge charger	17	4	227	2
porsche 911	7	4	5	234

```python
from sklearn.metrics import confusion_matrix

predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)

true_classes = test_generator.classes

confusion = confusion_matrix(true_classes, predicted_classes)
print('Matriz de Confusão:')
print(confusion)
```

```
63/63 [==============================] - 5s 84ms/step
Matriz de Confusão:
[[246   0   1   3]
 [ 12 237   0   1]
 [ 17   4 227   2]
 [  7   4   5 234]]
```

Bacharelado em Ciência da Computação
Disciplina: Inteligência Artificial



TRABALHO PRÁTICO SOBRE REDES NEURAIS CONVOLUCIONAIS

Discentes: João Gabriel Garcia de Sousa, Mateus Cesar Marques, Yuri Neres Macedo
Docente: Hugo Resende