Trabalho #2 - Rede residual e ajuste de modelo

Nesse trabalho você vai criar uma rede residual para realizar uma tarefa de classificação de múltiplas classes.

A tarefa consiste em classificar objetos do conjunto de dados CIFAR100

Esse trabalho é dividido nas seguintes etapas:

1. Explorar as imagens do conjunto de dados;
2. Construir e treinar uma rede residual para identificar o objeto mostrado na imagem;
3. Avaliar o desempenho da rede;
4. Ajustar o modelo para melhorar o desempenho.

Importação das principais bibliotecas

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
tf.__version__
```

{"type":"string"}

1. Conjunto de dados

O conjunto de dados CIFAR10 é composto por imagens coloridas de dimensão (32, 32, 3).

Esse conjunto de imagens é dedicado à classificçaõ multiclasse com 10 tipos de objetos.

Execute a célula abaixo para carregar o conjunto de dados CIFAR-10, que já está disponível diretamente no Keras.

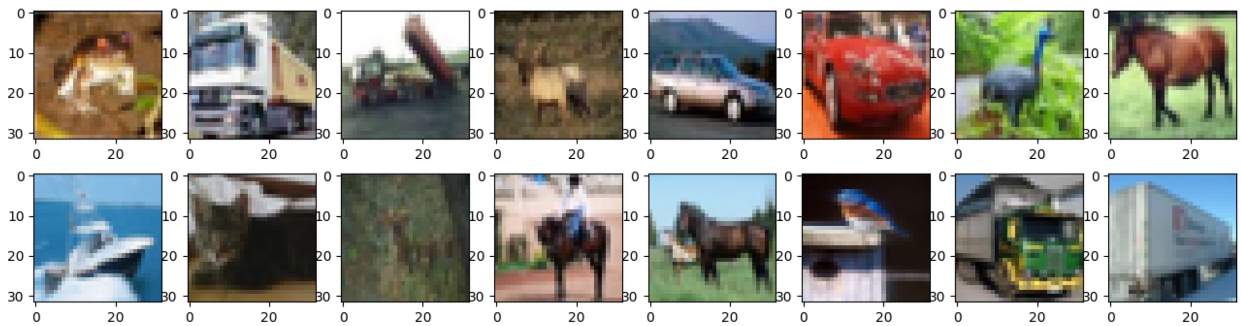
```
# Importar conjunto de dados
from tensorflow.keras.datasets import cifar10

# Carregar o conjunto de dados CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 6s 0us/step
```

Execute a célula abaixo para visualizar algumas imagens.

```
fig, axs = plt.subplots(2, 8, figsize=(16, 4))
index = 0
for i in range(2):
    for j in range(8):
```

```
        axs[i,j].imshow(x_train[index])
        index += 1
plt.show()
```



Exercício #1: Pré-processamento dos dados

Na célula abaixo crie um código para realizar o seguinte:

1. Normalizar as imagens de forma que os seus pixels sejam valores reais entre 0 e 1;
2. Codificação one-hot das saídas.

```
# PASRA VOCÊ FAZER: Normalização das imagens e codificação one-hot das
saídas
# Importa função to_categorical
from tensorflow.keras.utils import to_categorical

# Normalização das imagens
### COMECE AQUI #### (~2 linhas)
x_train_norm = x_train.astype('float32') / 255.0
x_test_norm = x_test.astype('float32') / 255.0
### TERMINE AQUI ###

# Converter as classes para one-hot encoding
### COMECE AQUI #### (~2 linhas)
y_train_hot = to_categorical(y_train, num_classes=10)
y_test_hot = to_categorical(y_test, num_classes=10)
### TERMINE AQUI ###

print(f"Dimensão do conjunto de treinamento: {x_train_norm.shape}")
print(f"Dimensão do conjunto de teste: {x_test_norm.shape}")
print(f"Dimensão das saídas de treinamento: {y_train_hot.shape}")
print(f"Dimensão das saídas de teste: {y_test_hot.shape}")
print('Valores máximos e mínimos das imagens de treinamento:',
np.max(x_train_norm), ',', np.min(x_train_norm))
print('Valores máximos e mínimos das imagens de teste:',
np.max(x_test_norm), ',',  np.min(x_test_norm))
```

```
Dimensão do conjunto de treinamento: (50000, 32, 32, 3)
Dimensão do conjunto de teste: (10000, 32, 32, 3)
Dimensão das saídas de treinamento: (50000, 10)
Dimensão das saídas de teste: (10000, 10)
Valores máximos e mínimos das imagens de treinamento: 1.0 , 0.0
Valores máximos e mínimos das imagens de teste: 1.0 , 0.0
```

Saída esperada:

```
Dimensão do conjunto de treinamento: (50000, 32, 32, 3)
Dimensão do conjunto de teste: (10000, 32, 32, 3)
Dimensão das saídas de treinamento: (50000, 10)
Dimensão das saídas de teste: (10000, 10)
Valores máximos e mínimos das imagens: 1.0 0.0
```

2. Configuração do modelo

Nesse trabalho você vai criar uma rede residual com camadas densas.

Uma rede residual é composta por blocos residuais. Na Figura 1, é mostrado um bloco residual.

ASa equações que impelementam esse bloco são as seguintes:

$$a^{[l+1]} = dense(a^{[l]})$$

$$z^{[l+2]} = dense(a^{[l]})$$

$$a^{[l+2]} = g^{[l+2]}(z^{[l+2]} + a^{[l]}) ¿$$

onde $dense$ é uma camada densa. Observe que a função de ativação da segunda camada densa do bloco somente é aplicada após a soma das ativações $a^{[l]}$ e dos estados $z^{[l+2]}$.

A camada `layers.Add()` do Keras realiza a soma de dois tensores.

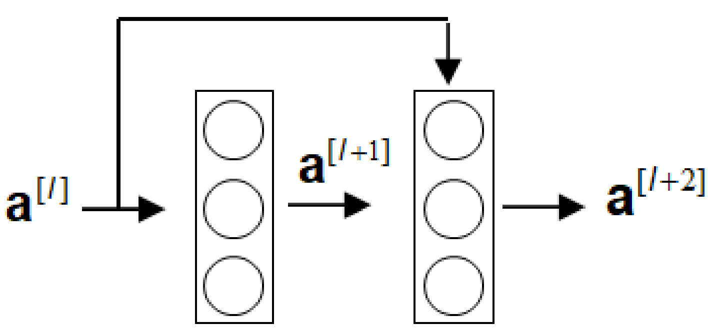


Figura 1 - Esquema de um bloco residual.

Exercício #2: Codificação do bloco residual

Na célula abaixo crie uma função que implementa o bloco residual mostrado na Figura 1. Use a função de ativação relu.

```
### PARA VOCÊ FAZER: Definir o bloco residual
# Importa classe de camadas e modelos
from tensorflow.keras import layers, models

# Importa função de ativação
from tensorflow.keras.activations import relu

def residual_block(x, units):
    # Primeira camada densa
    ### COMECE AQUI ### (1 linha)
    z1 = layers.Dense(units, activation=relu)(x)
    ### TERMINE AQUI ###

    # Segunda camada densa sem ativação
    ### COMECE AQUI ### (1 linha)
    z2 = layers.Dense(units, activation=None)(z1)
    ### TERMINE AQUI ###

    # Soma entrada com z2 com saída da segunda camada
    ### COMECE AQUI ### (1 linha)
    added = layers.Add()([x, z2])
    ### TERMINE AQUI ###

    # Aplica função de ativação
    ### COMECE AQUI ### (1 linha)
    a2 = layers.Activation('relu')(added)
    ### TERMINE AQUI ###

    return a2
```

Execute a célula abaixo para testar o seu bloco residual.

```
np.random.seed(3)
x = np.random.randn(3,5)
a2 = residual_block(x, 5)
print('x:', x, '\n')
print('a2:', a2)
```

```
x: [[ 1.78862847  0.43650985  0.09649747 -1.8634927  -0.2773882 ]
 [-0.35475898 -0.08274148 -0.62700068 -0.04381817 -0.47721803]
 [-1.31386475  0.88462238  0.88131804  1.70957306  0.05003364]]

a2: tf.Tensor(
[[1.0679691  1.1880854  0.         0.         0.42664248]
 [0.         0.34136277 0.         0.         0.        ]
```

```
 [0.         0.7346052  0.01420879 0.         0.        ]], shape=(3, 
5), dtype=float32)
```

Saída esperada:

```
x: [[ 1.78862847  0.43650985  0.09649747 -1.8634927  -0.2773882 ]
[-0.35475898 -0.08274148 -0.62700068 -0.04381817 -0.47721803]
[-1.31386475  0.88462238  0.88131804  1.70957306  0.05003364]]

a2: tf.Tensor(
[[1.6887243  0.5611134  0.5987834  0.         0.67116445]
[0.         0.03314844 0.         0.1769045  0.        ]
[0.3389045  0.16169035 0.         0.7670167  0.        ]], shape=(3, 
5), dtype=float32)
```

Exercício #3: Configuração da rede

A rede que será utilizada será composta por 2 blocos residuais, sendo que entre eles deve ter um camada densa para ajustar a dimensão dos dados.

Na célula abaixo crie uma função para gerar a rede residual. Essa função deve receber o seguinte:

1. Dimensão das imagens de entrada;
2. lista com número de unidades em cada bloco e cada camada da rede;
3. Número de classes para definir o número de unidades da camada de saída;
4. Exceto na camada de saída utiliza função de ativação relu;
5. Inclua camadas de dropout após cada camada densa e após cada bloco residual.

```python
### PARA VOCÊ FAZER: Definir a arquitetura do modelo

def create_residual_model(n1, n2, num_classes, input_shape=(32, 32, 
3)):
    ### COMECE AQUI ###
    # Camada de entrada
    inputs = layers.Input(shape=input_shape)

    # Camada para esticar a imagem (flattening)
    x = layers.Flatten()(inputs)

    # Primeira camada densa com dropout
    x = layers.Dense(n1, activation='relu')(x)
    x = layers.Dropout(0.05)(x)  # Dropout para regularização

    # Primeiro bloco residual
    x = residual_block(x, n1)
    x = layers.Dropout(0.05)(x)  # Dropout após o bloco residual

    # Camada densa intermediária para ajustar a dimensão
```

```
    x = layers.Dense(n2, activation='relu')(x)
    x = layers.Dropout(0.05)(x)  # Dropout após a camada intermediária

    # Segundo bloco residual
    x = residual_block(x, n2)
    x = layers.Dropout(0.05)(x)  # Dropout após o bloco residual

    # Camada de saída (número de classes)
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    # Construir o modelo
    model = models.Model(inputs=inputs, outputs=outputs)
    ### TERMINE AQUI ###

    return model
```

Para criar o modelo, utilize a função `create_residual_model()` com os seguintes parâmetros:

- n1 = 256;
- n2 = 128;
- num_classes = 10.

```
# PARA VOCE FAZER: Criar modelo

# Definir parâmetros
### PARA VOCE FAZER ### (3 linhas)
n1 = 256
n2 = 128
num_classes = 10
### TERMINE AQUI ###

# Criar modelo
### PARA VOCE FAZER ### (1 linha)
rna = create_residual_model(n1, n2, num_classes)
### TERMINE AQUI ###

# Resumo do modelo
rna.summary()
```

```
Model: "functional_2"
```

Layer (type) Connected to	Output Shape	Param #
input_layer_2 -	(None, 32, 32, 3)	0

(InputLayer)			
flatten_2 (Flatten)	(None, 3072)	0	input_layer_2[0][0]
dense_16 (Dense)	(None, 256)	786,688	flatten_2[0][0]
dropout_8 (Dropout)	(None, 256)	0	dense_16[0][0]
dense_17 (Dense)	(None, 256)	65,792	dropout_8[0][0]
dense_18 (Dense)	(None, 256)	65,792	dense_17[0][0]
add_5 (Add)	(None, 256)	0	dropout_8[0][0], dense_18[0][0]
activation_5 (Activation)	(None, 256)	0	add_5[0][0]
dropout_9 (Dropout)	(None, 256)	0	activation_5[0][0]
dense_19 (Dense)	(None, 128)	32,896	dropout_9[0][0]
dropout_10 (Dropout)	(None, 128)	0	dense_19[0][0]
dense_20 (Dense)	(None, 128)	16,512	dropout_10[0][0]

```
┤                                       ┤
 dense_21 (Dense)                       │ (None, 128)              │          16,512
 dense_20[0][0]                         │
├─────────────────────────────────────────┼──────────────────────────┼────────────────
┤                                       ┤
 add_6 (Add)                            │ (None, 128)              │               0
 dropout_10[0][0],                      │
                                        │                          │
 dense_21[0][0]                         │
├─────────────────────────────────────────┼──────────────────────────┼────────────────
┤                                       ┤
 activation_6 (Activation)              │ (None, 128)              │               0
 add_6[0][0]                            │
├─────────────────────────────────────────┼──────────────────────────┼────────────────
┤                                       ┤
 dropout_11 (Dropout)                   │ (None, 128)              │               0
 activation_6[0][0]                     │
├─────────────────────────────────────────┼──────────────────────────┼────────────────
┤                                       ┤
 dense_22 (Dense)                       │ (None, 10)               │           1,290
 dropout_11[0][0]                       │
└─────────────────────────────────────────┴──────────────────────────┴────────────────
┘                                       ┘

 Total params: 985,482 (3.76 MB)

 Trainable params: 985,482 (3.76 MB)

 Non-trainable params: 0 (0.00 B)
```

Saída esperada:

```
Model: "functional"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━
┓                                       ┓
 Layer (type)                           ┃ Output Shape             ┃         Param #
 Connected to                           ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━
┩                                       ┩
 input_layer_1                          │ (None, 32, 32, 3)        │               0
 -                                      │
 (InputLayer)                           │                          │
                                        │
├─────────────────────────────────────────┼──────────────────────────┼────────────────
┤                                       ┤
 flatten_1 (Flatten)                    │ (None, 3072)             │               0
 input_layer_1[0][0]                    │
├─────────────────────────────────────────┼──────────────────────────┼────────────────
┤                                       ┤
 dense_12 (Dense)                       │ (None, 256)              │         786,688
```

			flatten_1[0][0]
dense_13 (Dense)	(None, 256)	65,792	dense_12[0][0]
dense_14 (Dense)	(None, 256)	65,792	dense_13[0][0]
add_6 (Add)	(None, 256)	0	dense_12[0][0], dense_14[0][0]
activation_4 (Activation)	(None, 256)	0	add_6[0][0]
dense_15 (Dense)	(None, 128)	32,896	activation_4[0][0]
dense_16 (Dense)	(None, 128)	16,512	dense_15[0][0]
dense_17 (Dense)	(None, 128)	16,512	dense_16[0][0]
add_7 (Add)	(None, 128)	0	dense_15[0][0], dense_17[0][0]
activation_5 (Activation)	(None, 128)	0	add_7[0][0]
dense_18 (Dense)	(None, 10)	1,290	activation_5[0][0]

Total params: 985,482 (3.76 MB)

```
Trainable params: 985,482 (3.76 MB)
Non-trainable params: 0 (0.00 B)
```

3. Compilação e treinamento do modelo

Exercício #4: Compilação do modelo

Agora que o modelo foi criado, você precisa compilá-lo. Vamos usar a função de perda `categorical_crossentropy` para classificação multiclasse e o otimizador `Adam`.

```
# PARA VOCE FAZER: Compilar o modelo
from tensorflow.keras.optimizers import Adam
### COMECE AQUI ### (1 comando)

rna.compile(optimizer=Adam(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])

### TERMINE AQUI ###
```

Exercício #5: Treinar o modelo

Para treinar o modelo use 100 épocas e um lote de 256 exemplos.

```
# PARA VOCE FAZER: Treinar o modelo

### COMECE AQUI ### (1 comando)

history = rna.fit(x_train_norm, y_train_hot, epochs=100,
batch_size=256, validation_data=(x_test_norm, y_test_hot))

### TERMINE AQUI ###
```

```
Epoch 1/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 9s 22ms/step - accuracy: 0.2216 - loss:
2.1222 - val_accuracy: 0.3584 - val_loss: 1.7977
Epoch 2/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.3549 - loss:
1.8107 - val_accuracy: 0.4064 - val_loss: 1.6714
Epoch 3/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.3853 - loss:
1.7133 - val_accuracy: 0.4202 - val_loss: 1.6280
Epoch 4/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4109 - loss:
1.6478 - val_accuracy: 0.4471 - val_loss: 1.5649
Epoch 5/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4259 - loss:
1.6056 - val_accuracy: 0.4520 - val_loss: 1.5329
Epoch 6/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4459 - loss:
```

```
1.5593 - val_accuracy: 0.4646 - val_loss: 1.5050
Epoch 7/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4583 - loss:
1.5228 - val_accuracy: 0.4675 - val_loss: 1.4939
Epoch 8/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.4606 - loss:
1.5107 - val_accuracy: 0.4729 - val_loss: 1.4688
Epoch 9/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.4738 - loss:
1.4782 - val_accuracy: 0.4877 - val_loss: 1.4446
Epoch 10/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.4820 - loss:
1.4524 - val_accuracy: 0.4890 - val_loss: 1.4390
Epoch 11/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.4884 - loss:
1.4338 - val_accuracy: 0.4872 - val_loss: 1.4283
Epoch 12/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5022 - loss:
1.4013 - val_accuracy: 0.4985 - val_loss: 1.4055
Epoch 13/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5047 - loss:
1.3904 - val_accuracy: 0.5018 - val_loss: 1.3963
Epoch 14/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5135 - loss:
1.3688 - val_accuracy: 0.5007 - val_loss: 1.4006
Epoch 15/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5146 - loss:
1.3526 - val_accuracy: 0.4935 - val_loss: 1.4210
Epoch 16/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5210 - loss:
1.3462 - val_accuracy: 0.5071 - val_loss: 1.3837
Epoch 17/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5334 - loss:
1.3216 - val_accuracy: 0.5156 - val_loss: 1.3635
Epoch 18/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5381 - loss:
1.3026 - val_accuracy: 0.4975 - val_loss: 1.3958
Epoch 19/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5367 - loss:
1.3000 - val_accuracy: 0.5156 - val_loss: 1.3565
Epoch 20/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5410 - loss:
1.2809 - val_accuracy: 0.5137 - val_loss: 1.3529
Epoch 21/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5444 - loss:
1.2730 - val_accuracy: 0.5146 - val_loss: 1.3722
Epoch 22/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5518 - loss:
1.2606 - val_accuracy: 0.5165 - val_loss: 1.3473
```

```
Epoch 23/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5566 - loss:
1.2473 - val_accuracy: 0.5211 - val_loss: 1.3414
Epoch 24/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5568 - loss:
1.2410 - val_accuracy: 0.5268 - val_loss: 1.3279
Epoch 25/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5680 - loss:
1.2117 - val_accuracy: 0.5240 - val_loss: 1.3352
Epoch 26/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5755 - loss:
1.1926 - val_accuracy: 0.5327 - val_loss: 1.3274
Epoch 27/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5716 - loss:
1.1955 - val_accuracy: 0.5248 - val_loss: 1.3211
Epoch 28/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5779 - loss:
1.1858 - val_accuracy: 0.5322 - val_loss: 1.3264
Epoch 29/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.5811 - loss:
1.1659 - val_accuracy: 0.5282 - val_loss: 1.3288
Epoch 30/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5861 - loss:
1.1555 - val_accuracy: 0.5308 - val_loss: 1.3247
Epoch 31/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5939 - loss:
1.1350 - val_accuracy: 0.5275 - val_loss: 1.3240
Epoch 32/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6010 - loss:
1.1145 - val_accuracy: 0.5255 - val_loss: 1.3301
Epoch 33/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5978 - loss:
1.1293 - val_accuracy: 0.5350 - val_loss: 1.3189
Epoch 34/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6006 - loss:
1.1224 - val_accuracy: 0.5313 - val_loss: 1.3327
Epoch 35/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6058 - loss:
1.0976 - val_accuracy: 0.5260 - val_loss: 1.3464
Epoch 36/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6107 - loss:
1.0905 - val_accuracy: 0.5313 - val_loss: 1.3398
Epoch 37/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6114 - loss:
1.0840 - val_accuracy: 0.5323 - val_loss: 1.3244
Epoch 38/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6198 - loss:
1.0638 - val_accuracy: 0.5399 - val_loss: 1.3084
Epoch 39/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6161 - loss:
1.0740 - val_accuracy: 0.5372 - val_loss: 1.3310
Epoch 40/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6212 - loss:
1.0579 - val_accuracy: 0.5361 - val_loss: 1.3310
Epoch 41/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6221 - loss:
1.0547 - val_accuracy: 0.5357 - val_loss: 1.3213
Epoch 42/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6324 - loss:
1.0331 - val_accuracy: 0.5403 - val_loss: 1.3179
Epoch 43/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6280 - loss:
1.0308 - val_accuracy: 0.5378 - val_loss: 1.3256
Epoch 44/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6341 - loss:
1.0221 - val_accuracy: 0.5319 - val_loss: 1.3352
Epoch 45/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6348 - loss:
1.0232 - val_accuracy: 0.5278 - val_loss: 1.3520
Epoch 46/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6400 - loss:
1.0059 - val_accuracy: 0.5348 - val_loss: 1.3295
Epoch 47/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6426 - loss:
1.0009 - val_accuracy: 0.5355 - val_loss: 1.3575
Epoch 48/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6410 - loss:
0.9969 - val_accuracy: 0.5384 - val_loss: 1.3512
Epoch 49/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6525 - loss:
0.9729 - val_accuracy: 0.5448 - val_loss: 1.3311
Epoch 50/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6503 - loss:
0.9719 - val_accuracy: 0.5425 - val_loss: 1.3460
Epoch 51/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6497 - loss:
0.9718 - val_accuracy: 0.5411 - val_loss: 1.3596
Epoch 52/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.6575 - loss:
0.9595 - val_accuracy: 0.5400 - val_loss: 1.3401
Epoch 53/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6599 - loss:
0.9481 - val_accuracy: 0.5394 - val_loss: 1.3537
Epoch 54/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6623 - loss:
0.9462 - val_accuracy: 0.5446 - val_loss: 1.3389
Epoch 55/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6665 - loss:
```

```
0.9262 - val_accuracy: 0.5421 - val_loss: 1.3584
Epoch 56/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6636 - loss:
0.9369 - val_accuracy: 0.5397 - val_loss: 1.3580
Epoch 57/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6701 - loss:
0.9212 - val_accuracy: 0.5398 - val_loss: 1.3602
Epoch 58/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6732 - loss:
0.9108 - val_accuracy: 0.5443 - val_loss: 1.3771
Epoch 59/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6739 - loss:
0.9000 - val_accuracy: 0.5445 - val_loss: 1.3752
Epoch 60/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6751 - loss:
0.8996 - val_accuracy: 0.5439 - val_loss: 1.3699
Epoch 61/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6832 - loss:
0.8886 - val_accuracy: 0.5357 - val_loss: 1.3857
Epoch 62/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6860 - loss:
0.8737 - val_accuracy: 0.5477 - val_loss: 1.3546
Epoch 63/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6848 - loss:
0.8736 - val_accuracy: 0.5434 - val_loss: 1.4229
Epoch 64/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6845 - loss:
0.8707 - val_accuracy: 0.5432 - val_loss: 1.3992
Epoch 65/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6922 - loss:
0.8557 - val_accuracy: 0.5433 - val_loss: 1.3966
Epoch 66/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6962 - loss:
0.8477 - val_accuracy: 0.5387 - val_loss: 1.4131
Epoch 67/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6968 - loss:
0.8459 - val_accuracy: 0.5382 - val_loss: 1.4099
Epoch 68/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6990 - loss:
0.8390 - val_accuracy: 0.5407 - val_loss: 1.3700
Epoch 69/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7004 - loss:
0.8307 - val_accuracy: 0.5315 - val_loss: 1.4172
Epoch 70/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7021 - loss:
0.8287 - val_accuracy: 0.5456 - val_loss: 1.4043
Epoch 71/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7048 - loss:
0.8183 - val_accuracy: 0.5448 - val_loss: 1.4268
```

```
Epoch 72/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7102 - loss:
0.8107 - val_accuracy: 0.5465 - val_loss: 1.4136
Epoch 73/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.7139 - loss:
0.8031 - val_accuracy: 0.5409 - val_loss: 1.4390
Epoch 74/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7116 - loss:
0.8015 - val_accuracy: 0.5394 - val_loss: 1.4428
Epoch 75/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7166 - loss:
0.7928 - val_accuracy: 0.5478 - val_loss: 1.4224
Epoch 76/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7190 - loss:
0.7818 - val_accuracy: 0.5447 - val_loss: 1.4453
Epoch 77/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7193 - loss:
0.7804 - val_accuracy: 0.5451 - val_loss: 1.4267
Epoch 78/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7226 - loss:
0.7762 - val_accuracy: 0.5428 - val_loss: 1.4541
Epoch 79/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7185 - loss:
0.7817 - val_accuracy: 0.5460 - val_loss: 1.4492
Epoch 80/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7257 - loss:
0.7594 - val_accuracy: 0.5439 - val_loss: 1.4584
Epoch 81/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7298 - loss:
0.7522 - val_accuracy: 0.5444 - val_loss: 1.4843
Epoch 82/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7352 - loss:
0.7388 - val_accuracy: 0.5451 - val_loss: 1.4779
Epoch 83/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7273 - loss:
0.7486 - val_accuracy: 0.5415 - val_loss: 1.4683
Epoch 84/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7411 - loss:
0.7340 - val_accuracy: 0.5377 - val_loss: 1.4928
Epoch 85/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7346 - loss:
0.7369 - val_accuracy: 0.5358 - val_loss: 1.4959
Epoch 86/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7376 - loss:
0.7324 - val_accuracy: 0.5386 - val_loss: 1.5041
Epoch 87/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7441 - loss:
0.7156 - val_accuracy: 0.5397 - val_loss: 1.5040
Epoch 88/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7439 - loss:
0.7170 - val_accuracy: 0.5327 - val_loss: 1.5372
Epoch 89/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7412 - loss:
0.7184 - val_accuracy: 0.5386 - val_loss: 1.5348
Epoch 90/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7437 - loss:
0.7087 - val_accuracy: 0.5415 - val_loss: 1.5012
Epoch 91/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7460 - loss:
0.7030 - val_accuracy: 0.5402 - val_loss: 1.5254
Epoch 92/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7535 - loss:
0.6952 - val_accuracy: 0.5447 - val_loss: 1.5334
Epoch 93/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7561 - loss:
0.6809 - val_accuracy: 0.5418 - val_loss: 1.5611
Epoch 94/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.7541 - loss:
0.6818 - val_accuracy: 0.5447 - val_loss: 1.5259
Epoch 95/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7637 - loss:
0.6684 - val_accuracy: 0.5397 - val_loss: 1.5792
Epoch 96/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7551 - loss:
0.6866 - val_accuracy: 0.5402 - val_loss: 1.5546
Epoch 97/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7646 - loss:
0.6568 - val_accuracy: 0.5433 - val_loss: 1.5585
Epoch 98/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7615 - loss:
0.6602 - val_accuracy: 0.5420 - val_loss: 1.5565
Epoch 99/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7663 - loss:
0.6496 - val_accuracy: 0.5355 - val_loss: 1.5742
Epoch 100/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7678 - loss:
0.6511 - val_accuracy: 0.5422 - val_loss: 1.5975
```

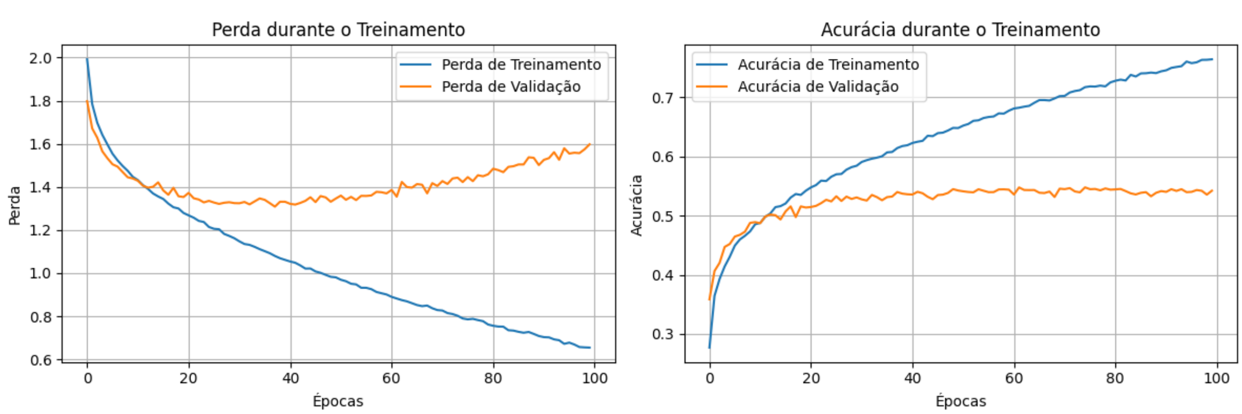
Saída esperada:

```
Epoch 1/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.2288 - loss:
2.1627 - val_accuracy: 0.3783 - val_loss: 1.7425
Epoch 2/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.3720 - loss:
1.7419 - val_accuracy: 0.4168 - val_loss: 1.6358
.
.
```

```
.
Epoch 99/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9347 - loss:
0.1806 - val_accuracy: 0.4800 - val_loss: 4.0803
Epoch 100/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9350 - loss:
0.1817 - val_accuracy: 0.4942 - val_loss: 4.1091
```

Execute a célula abaixo para vizualizar os gráficos do processo de treinamento.

```
# Plotar a perda e acurácia durante o treinamento
plt.figure(figsize=(12, 4))

# Plotando a perda
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Perda de Treinamento')
plt.plot(history.history['val_loss'], label='Perda de Validação')
plt.title('Perda durante o Treinamento')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.grid()
plt.legend()

# Plotando a acurácia
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Acurácia de Treinamento')
plt.plot(history.history['val_accuracy'], label='Acurácia de
Validação')
plt.title('Acurácia durante o Treinamento')
plt.xlabel('Épocas')
plt.ylabel('Acurácia')
plt.grid()
plt.legend()

plt.tight_layout()
plt.show()
```



Exercício #6: Avaliar o modelo

Use o método evaluate e calcule a função de custo e a métrica para os dados de treinamento e teste.

```
### PARA VOCE FAZER: Avaliar o modelo

### COMECE AQUI ### (2 linhas)

train_loss, train_accuracy = rna.evaluate(x_train_norm, y_train_hot,
verbose=0)
test_loss, test_accuracy = rna.evaluate(x_test_norm, y_test_hot,
verbose=0)

### TERMINE AQUI ###
print(f"Função de custo no conjunto de treinamento: {train_loss:.4f}")
print(f"Acurácia no conjunto de treinamento: {train_accuracy:.4f}")
print(f"Função de custo no conjunto de teste: {test_loss:.4f}")
print(f"Acurácia no conjunto de teste: {test_accuracy:.4f}")

Função de custo no conjunto de treinamento: 0.4840
Acurácia no conjunto de treinamento: 0.8343
Função de custo no conjunto de teste: 1.5975
Acurácia no conjunto de teste: 0.5422
```

Saída esperada:

```
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9438 - loss:
0.1639
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.4957 - loss:
4.0715
```

4. Ajuste do modelo

Você agora tem um modelo básico de rede residual para classificação de imagens no conjunto CIFAR100. Por meio da adição de blocos residuais, o modelo pode ser mais profundo sem enfrentar problemas de degradação do desempenho.

Com base nesse modelo, você pode fazer experimentos adicionais para melhorar a performance, como ajustar a arquitetura, adicionar camadas de normalização, ou utilizar técnicas de data augmentation.

Exercício #7: Ajustar o modelo para obter resultados melhores

Nesse exercício você deve fazer ajustes no modelo para melhorar o seu desempenho. As possíveis modificações são: adicionar mais blocos residuais, mudar o número de unidades nos blocos e nas camadas densas, ou tentar diferentes técnicas de regularização (dropout, L2 regularization etc.) Além disso, é possível testar diferentes algoritmos de otimização e taxas de aprendizado.

Implemente as seguintes modificações no modelo:

1. Aumentar número de blocos residuais e aumentar número de unidades nas camadas;
2. Incluir camadas de dropout no modelo maior do item (1);
3. Retirar as camadas de dropout e aplicar regularização L2 no modelo maior do item(1).

Para cada modificação você deve apresentar o novo modelo, a compilação, o treinamento e a avaliação.

A sua avaliação nesse exercício depende dos resultados de exatidão nos dados de teste.

Analise os resultados do ajuste do modelo.

#1. Aumentar o número de blocos residuais e unidades nas camadas Novo modelo com mais blocos e unidades:

```
def create_residual_model_v1(n1, n2, n3, num_classes, input_shape=(32,
32, 3)):
    inputs = layers.Input(shape=input_shape)
    x = layers.Flatten()(inputs)

    # Bloco 1
    x = layers.Dense(n1, activation='relu')(x)
    x = residual_block(x, n1)

    # Bloco 2
    x = layers.Dense(n2, activation='relu')(x)
    x = residual_block(x, n2)

    # Bloco 3 (novo bloco)
    x = layers.Dense(n3, activation='relu')(x)
    x = residual_block(x, n3)

    # Camada de saída
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = models.Model(inputs=inputs, outputs=outputs)
    return model

# Definir parâmetros
n1, n2, n3 = 512, 256, 128  # Mais unidades e um bloco adicional
num_classes = 10

# Criar e compilar o modelo
rna_v1 = create_residual_model_v1(n1, n2, n3, num_classes)
rna_v1.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Treinar o modelo
history_v1 = rna_v1.fit(x_train_norm, y_train_hot, epochs=100,
batch_size=256, validation_data=(x_test_norm, y_test_hot))
```

```python
# Avaliar o modelo
train_loss_v1, train_accuracy_v1 = rna_v1.evaluate(x_train_norm,
y_train_hot)
test_loss_v1, test_accuracy_v1 = rna_v1.evaluate(x_test_norm,
y_test_hot)

print(f"Modelo v1 - Acurácia no teste: {test_accuracy_v1:.4f}")
```

```
Epoch 1/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 8s 19ms/step - accuracy: 0.2100 - loss:
2.2413 - val_accuracy: 0.3657 - val_loss: 1.7857
Epoch 2/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - accuracy: 0.3697 - loss:
1.7477 - val_accuracy: 0.4147 - val_loss: 1.6424
Epoch 3/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.4072 - loss:
1.6539 - val_accuracy: 0.4158 - val_loss: 1.6270
Epoch 4/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4442 - loss:
1.5544 - val_accuracy: 0.4226 - val_loss: 1.6146
Epoch 5/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4534 - loss:
1.5232 - val_accuracy: 0.4726 - val_loss: 1.4742
Epoch 6/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4774 - loss:
1.4619 - val_accuracy: 0.4701 - val_loss: 1.4795
Epoch 7/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4881 - loss:
1.4307 - val_accuracy: 0.4746 - val_loss: 1.4923
Epoch 8/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5044 - loss:
1.3852 - val_accuracy: 0.4897 - val_loss: 1.4457
Epoch 9/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5209 - loss:
1.3443 - val_accuracy: 0.4869 - val_loss: 1.4342
Epoch 10/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5289 - loss:
1.3141 - val_accuracy: 0.4959 - val_loss: 1.4251
Epoch 11/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.5404 - loss:
1.2922 - val_accuracy: 0.4867 - val_loss: 1.4740
Epoch 12/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.5468 - loss:
1.2562 - val_accuracy: 0.5058 - val_loss: 1.3985
Epoch 13/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5668 - loss:
1.2152 - val_accuracy: 0.5069 - val_loss: 1.3978
Epoch 14/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5750 - loss:
```

```
1.1874 - val_accuracy: 0.5100 - val_loss: 1.3998
Epoch 15/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5874 - loss:
1.1503 - val_accuracy: 0.5103 - val_loss: 1.3968
Epoch 16/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6055 - loss:
1.1052 - val_accuracy: 0.5111 - val_loss: 1.4231
Epoch 17/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6183 - loss:
1.0637 - val_accuracy: 0.5199 - val_loss: 1.4016
Epoch 18/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6294 - loss:
1.0181 - val_accuracy: 0.5200 - val_loss: 1.4154
Epoch 19/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6451 - loss:
0.9823 - val_accuracy: 0.5167 - val_loss: 1.4572
Epoch 20/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6646 - loss:
0.9379 - val_accuracy: 0.5189 - val_loss: 1.4828
Epoch 21/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6754 - loss:
0.9015 - val_accuracy: 0.5214 - val_loss: 1.4828
Epoch 22/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.6971 - loss:
0.8368 - val_accuracy: 0.5233 - val_loss: 1.4836
Epoch 23/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.7051 - loss:
0.8172 - val_accuracy: 0.5033 - val_loss: 1.5928
Epoch 24/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7281 - loss:
0.7589 - val_accuracy: 0.5113 - val_loss: 1.6107
Epoch 25/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7414 - loss:
0.7128 - val_accuracy: 0.5094 - val_loss: 1.7272
Epoch 26/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7544 - loss:
0.6874 - val_accuracy: 0.5055 - val_loss: 1.7338
Epoch 27/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7733 - loss:
0.6327 - val_accuracy: 0.5038 - val_loss: 1.7784
Epoch 28/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7858 - loss:
0.5987 - val_accuracy: 0.5057 - val_loss: 1.8776
Epoch 29/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7882 - loss:
0.5864 - val_accuracy: 0.5034 - val_loss: 1.8736
Epoch 30/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7994 - loss:
0.5546 - val_accuracy: 0.5022 - val_loss: 1.9746
```

```
Epoch 31/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.8177 - loss:
0.5075 - val_accuracy: 0.5007 - val_loss: 2.0333
Epoch 32/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8264 - loss:
0.4848 - val_accuracy: 0.5094 - val_loss: 2.0333
Epoch 33/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8446 - loss:
0.4363 - val_accuracy: 0.5133 - val_loss: 2.1276
Epoch 34/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8487 - loss:
0.4286 - val_accuracy: 0.5003 - val_loss: 2.2313
Epoch 35/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8473 - loss:
0.4281 - val_accuracy: 0.4976 - val_loss: 2.2750
Epoch 36/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8714 - loss:
0.3668 - val_accuracy: 0.5069 - val_loss: 2.3478
Epoch 37/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8744 - loss:
0.3593 - val_accuracy: 0.5003 - val_loss: 2.4153
Epoch 38/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8801 - loss:
0.3433 - val_accuracy: 0.5016 - val_loss: 2.4916
Epoch 39/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8881 - loss:
0.3180 - val_accuracy: 0.4963 - val_loss: 2.5756
Epoch 40/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.8807 - loss:
0.3369 - val_accuracy: 0.5056 - val_loss: 2.7043
Epoch 41/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.8867 - loss:
0.3266 - val_accuracy: 0.4933 - val_loss: 2.5062
Epoch 42/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8923 - loss:
0.3047 - val_accuracy: 0.4945 - val_loss: 2.7585
Epoch 43/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8922 - loss:
0.3022 - val_accuracy: 0.4929 - val_loss: 2.9056
Epoch 44/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8969 - loss:
0.2933 - val_accuracy: 0.4937 - val_loss: 2.8504
Epoch 45/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9112 - loss:
0.2507 - val_accuracy: 0.4936 - val_loss: 2.8637
Epoch 46/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9012 - loss:
0.2764 - val_accuracy: 0.4967 - val_loss: 2.8733
Epoch 47/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9164 - loss:
0.2388 - val_accuracy: 0.4996 - val_loss: 2.8778
Epoch 48/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9142 - loss:
0.2494 - val_accuracy: 0.4909 - val_loss: 2.9809
Epoch 49/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9203 - loss:
0.2248 - val_accuracy: 0.4908 - val_loss: 3.0490
Epoch 50/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9194 - loss:
0.2317 - val_accuracy: 0.4862 - val_loss: 3.1615
Epoch 51/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9247 - loss:
0.2119 - val_accuracy: 0.5005 - val_loss: 2.9876
Epoch 52/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9310 - loss:
0.2038 - val_accuracy: 0.4950 - val_loss: 2.9873
Epoch 53/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9313 - loss:
0.2031 - val_accuracy: 0.5000 - val_loss: 3.2586
Epoch 54/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9301 - loss:
0.2016 - val_accuracy: 0.4976 - val_loss: 3.0739
Epoch 55/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9255 - loss:
0.2103 - val_accuracy: 0.4865 - val_loss: 3.1832
Epoch 56/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9269 - loss:
0.2128 - val_accuracy: 0.4830 - val_loss: 3.4391
Epoch 57/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9180 - loss:
0.2382 - val_accuracy: 0.4932 - val_loss: 3.2755
Epoch 58/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9406 - loss:
0.1762 - val_accuracy: 0.4863 - val_loss: 3.2207
Epoch 59/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9360 - loss:
0.1839 - val_accuracy: 0.4897 - val_loss: 3.5350
Epoch 60/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9303 - loss:
0.2044 - val_accuracy: 0.4935 - val_loss: 3.2941
Epoch 61/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9322 - loss:
0.1988 - val_accuracy: 0.4921 - val_loss: 3.2609
Epoch 62/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9466 - loss:
0.1496 - val_accuracy: 0.4890 - val_loss: 3.4327
Epoch 63/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9424 - loss:
```

```
0.1736 - val_accuracy: 0.5001 - val_loss: 3.4180
Epoch 64/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9431 - loss:
0.1625 - val_accuracy: 0.5013 - val_loss: 3.4691
Epoch 65/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9386 - loss:
0.1816 - val_accuracy: 0.4927 - val_loss: 3.5974
Epoch 66/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9449 - loss:
0.1641 - val_accuracy: 0.4917 - val_loss: 3.4941
Epoch 67/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9512 - loss:
0.1455 - val_accuracy: 0.4930 - val_loss: 3.4760
Epoch 68/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9388 - loss:
0.1801 - val_accuracy: 0.4862 - val_loss: 3.7142
Epoch 69/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9336 - loss:
0.1928 - val_accuracy: 0.4914 - val_loss: 3.6084
Epoch 70/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9468 - loss:
0.1571 - val_accuracy: 0.4897 - val_loss: 3.4872
Epoch 71/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9490 - loss:
0.1494 - val_accuracy: 0.4931 - val_loss: 3.5106
Epoch 72/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9461 - loss:
0.1578 - val_accuracy: 0.4934 - val_loss: 3.6673
Epoch 73/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9490 - loss:
0.1459 - val_accuracy: 0.4905 - val_loss: 3.7738
Epoch 74/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9535 - loss:
0.1374 - val_accuracy: 0.4986 - val_loss: 3.5972
Epoch 75/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9575 - loss:
0.1267 - val_accuracy: 0.4953 - val_loss: 3.9015
Epoch 76/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9546 - loss:
0.1343 - val_accuracy: 0.4868 - val_loss: 3.7815
Epoch 77/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9466 - loss:
0.1579 - val_accuracy: 0.4913 - val_loss: 3.6313
Epoch 78/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9448 - loss:
0.1689 - val_accuracy: 0.4904 - val_loss: 3.8423
Epoch 79/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9568 - loss:
0.1277 - val_accuracy: 0.4884 - val_loss: 3.7795
```

```
Epoch 80/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9488 - loss:
0.1459 - val_accuracy: 0.4991 - val_loss: 3.8126
Epoch 81/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9443 - loss:
0.1655 - val_accuracy: 0.4954 - val_loss: 3.8007
Epoch 82/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9591 - loss:
0.1203 - val_accuracy: 0.4917 - val_loss: 3.7763
Epoch 83/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.9559 - loss:
0.1319 - val_accuracy: 0.4913 - val_loss: 3.6443
Epoch 84/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9517 - loss:
0.1404 - val_accuracy: 0.4957 - val_loss: 3.9299
Epoch 85/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9598 - loss:
0.1147 - val_accuracy: 0.4991 - val_loss: 3.6427
Epoch 86/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9610 - loss:
0.1200 - val_accuracy: 0.4928 - val_loss: 3.7454
Epoch 87/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9557 - loss:
0.1352 - val_accuracy: 0.4895 - val_loss: 3.9916
Epoch 88/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9504 - loss:
0.1473 - val_accuracy: 0.4969 - val_loss: 3.8006
Epoch 89/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9635 - loss:
0.1072 - val_accuracy: 0.4930 - val_loss: 3.7527
Epoch 90/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9679 - loss:
0.1011 - val_accuracy: 0.4969 - val_loss: 4.0193
Epoch 91/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9619 - loss:
0.1181 - val_accuracy: 0.4981 - val_loss: 3.8934
Epoch 92/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9564 - loss:
0.1275 - val_accuracy: 0.4979 - val_loss: 3.8857
Epoch 93/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9635 - loss:
0.1116 - val_accuracy: 0.4894 - val_loss: 3.7400
Epoch 94/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.9624 - loss:
0.1114 - val_accuracy: 0.4936 - val_loss: 3.8450
Epoch 95/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9637 - loss:
0.1076 - val_accuracy: 0.4942 - val_loss: 3.8617
Epoch 96/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9606 - loss:
0.1167 - val_accuracy: 0.5000 - val_loss: 3.8527
Epoch 97/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9570 - loss:
0.1291 - val_accuracy: 0.4924 - val_loss: 3.9578
Epoch 98/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9582 - loss:
0.1241 - val_accuracy: 0.4969 - val_loss: 4.0638
Epoch 99/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9580 - loss:
0.1252 - val_accuracy: 0.4857 - val_loss: 3.9313
Epoch 100/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9639 - loss:
0.1148 - val_accuracy: 0.4996 - val_loss: 4.0057
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9638 - loss:
0.1113
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.5028 - loss:
4.0045
Modelo v1 - Acurácia no teste: 0.4996
```

#2. Incluir camadas de dropout no modelo maior Novo modelo com dropout:

```python
def create_residual_model_v2(n1, n2, n3, num_classes, input_shape=(32,
32, 3)):
    inputs = layers.Input(shape=input_shape)
    x = layers.Flatten()(inputs)

    # Bloco 1 com dropout
    x = layers.Dense(n1, activation='relu')(x)
    x = layers.Dropout(0.3)(x)
    x = residual_block(x, n1)

    # Bloco 2 com dropout
    x = layers.Dense(n2, activation='relu')(x)
    x = layers.Dropout(0.3)(x)
    x = residual_block(x, n2)

    # Bloco 3 com dropout
    x = layers.Dense(n3, activation='relu')(x)
    x = layers.Dropout(0.3)(x)
    x = residual_block(x, n3)

    # Camada de saída
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = models.Model(inputs=inputs, outputs=outputs)
    return model

# Criar e compilar o modelo com dropout
rna_v2 = create_residual_model_v2(n1, n2, n3, num_classes)
```

```python
rna_v2.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Treinar o modelo
history_v2 = rna_v2.fit(x_train_norm, y_train_hot, epochs=100,
batch_size=256, validation_data=(x_test_norm, y_test_hot))

# Avaliar o modelo
train_loss_v2, train_accuracy_v2 = rna_v2.evaluate(x_train_norm,
y_train_hot)
test_loss_v2, test_accuracy_v2 = rna_v2.evaluate(x_test_norm,
y_test_hot)

print(f"Modelo v2 - Acurácia no teste: {test_accuracy_v2:.4f}")
```

```
Epoch 1/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 10s 22ms/step - accuracy: 0.1567 - loss:
2.2834 - val_accuracy: 0.2768 - val_loss: 1.9227
Epoch 2/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - accuracy: 0.2738 - loss:
1.9593 - val_accuracy: 0.3289 - val_loss: 1.8546
Epoch 3/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.2937 - loss:
1.9009 - val_accuracy: 0.3151 - val_loss: 1.8740
Epoch 4/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3117 - loss:
1.8618 - val_accuracy: 0.3636 - val_loss: 1.8081
Epoch 5/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3210 - loss:
1.8447 - val_accuracy: 0.3706 - val_loss: 1.7755
Epoch 6/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3353 - loss:
1.8175 - val_accuracy: 0.3759 - val_loss: 1.7764
Epoch 7/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3397 - loss:
1.8099 - val_accuracy: 0.3702 - val_loss: 1.7823
Epoch 8/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3386 - loss:
1.7975 - val_accuracy: 0.3749 - val_loss: 1.7877
Epoch 9/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3472 - loss:
1.7734 - val_accuracy: 0.3710 - val_loss: 1.7866
Epoch 10/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3588 - loss:
1.7577 - val_accuracy: 0.3816 - val_loss: 1.7709
Epoch 11/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.3650 - loss:
1.7372 - val_accuracy: 0.3982 - val_loss: 1.7472
Epoch 12/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.3772 - loss:
```

```
1.7213 - val_accuracy: 0.4045 - val_loss: 1.7065
Epoch 13/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.3767 - loss:
1.7134 - val_accuracy: 0.4002 - val_loss: 1.7127
Epoch 14/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3816 - loss:
1.7061 - val_accuracy: 0.4049 - val_loss: 1.7331
Epoch 15/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3836 - loss:
1.7002 - val_accuracy: 0.4112 - val_loss: 1.7192
Epoch 16/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3904 - loss:
1.6812 - val_accuracy: 0.3917 - val_loss: 1.7498
Epoch 17/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3866 - loss:
1.6809 - val_accuracy: 0.4180 - val_loss: 1.6689
Epoch 18/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3973 - loss:
1.6634 - val_accuracy: 0.4283 - val_loss: 1.6533
Epoch 19/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4016 - loss:
1.6540 - val_accuracy: 0.4010 - val_loss: 1.7424
Epoch 20/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4000 - loss:
1.6564 - val_accuracy: 0.3982 - val_loss: 1.7183
Epoch 21/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.4029 - loss:
1.6553 - val_accuracy: 0.4116 - val_loss: 1.6980
Epoch 22/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4030 - loss:
1.6531 - val_accuracy: 0.3997 - val_loss: 1.7169
Epoch 23/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.4057 - loss:
1.6488 - val_accuracy: 0.4018 - val_loss: 1.7340
Epoch 24/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4060 - loss:
1.6298 - val_accuracy: 0.4060 - val_loss: 1.7343
Epoch 25/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4135 - loss:
1.6209 - val_accuracy: 0.3989 - val_loss: 1.7260
Epoch 26/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4124 - loss:
1.6270 - val_accuracy: 0.4015 - val_loss: 1.6924
Epoch 27/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4135 - loss:
1.6184 - val_accuracy: 0.4040 - val_loss: 1.7087
Epoch 28/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4241 - loss:
1.6080 - val_accuracy: 0.4308 - val_loss: 1.6402
```

```
Epoch 29/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4203 - loss: 
1.6059 - val_accuracy: 0.4282 - val_loss: 1.6771
Epoch 30/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4229 - loss: 
1.6050 - val_accuracy: 0.4218 - val_loss: 1.6757
Epoch 31/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4258 - loss: 
1.5894 - val_accuracy: 0.4161 - val_loss: 1.6982
Epoch 32/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4293 - loss: 
1.5864 - val_accuracy: 0.4243 - val_loss: 1.6527
Epoch 33/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4256 - loss: 
1.5897 - val_accuracy: 0.4072 - val_loss: 1.7051
Epoch 34/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4310 - loss: 
1.5832 - val_accuracy: 0.4314 - val_loss: 1.6632
Epoch 35/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4351 - loss: 
1.5762 - val_accuracy: 0.4110 - val_loss: 1.7113
Epoch 36/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4327 - loss: 
1.5743 - val_accuracy: 0.4103 - val_loss: 1.6962
Epoch 37/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4276 - loss: 
1.5838 - val_accuracy: 0.4208 - val_loss: 1.6918
Epoch 38/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4328 - loss: 
1.5807 - val_accuracy: 0.4347 - val_loss: 1.6549
Epoch 39/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4424 - loss: 
1.5585 - val_accuracy: 0.4185 - val_loss: 1.6812
Epoch 40/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4379 - loss: 
1.5651 - val_accuracy: 0.4265 - val_loss: 1.6717
Epoch 41/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4412 - loss: 
1.5562 - val_accuracy: 0.4140 - val_loss: 1.6935
Epoch 42/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4379 - loss: 
1.5642 - val_accuracy: 0.4124 - val_loss: 1.7028
Epoch 43/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4395 - loss: 
1.5527 - val_accuracy: 0.3957 - val_loss: 1.7335
Epoch 44/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.4378 - loss: 
1.5607 - val_accuracy: 0.4051 - val_loss: 1.7323
Epoch 45/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4399 - loss:
1.5572 - val_accuracy: 0.4037 - val_loss: 1.7040
Epoch 46/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4408 - loss:
1.5555 - val_accuracy: 0.4283 - val_loss: 1.6512
Epoch 47/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4463 - loss:
1.5452 - val_accuracy: 0.4317 - val_loss: 1.6671
Epoch 48/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4515 - loss:
1.5403 - val_accuracy: 0.4211 - val_loss: 1.6925
Epoch 49/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4468 - loss:
1.5363 - val_accuracy: 0.4196 - val_loss: 1.6719
Epoch 50/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4500 - loss:
1.5327 - val_accuracy: 0.4238 - val_loss: 1.6556
Epoch 51/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4480 - loss:
1.5330 - val_accuracy: 0.4225 - val_loss: 1.6682
Epoch 52/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.4502 - loss:
1.5327 - val_accuracy: 0.4168 - val_loss: 1.6857
Epoch 53/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4456 - loss:
1.5454 - val_accuracy: 0.3942 - val_loss: 1.7652
Epoch 54/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4514 - loss:
1.5327 - val_accuracy: 0.4301 - val_loss: 1.6703
Epoch 55/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4567 - loss:
1.5165 - val_accuracy: 0.4147 - val_loss: 1.6990
Epoch 56/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4533 - loss:
1.5271 - val_accuracy: 0.4331 - val_loss: 1.6691
Epoch 57/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4563 - loss:
1.5122 - val_accuracy: 0.4350 - val_loss: 1.6427
Epoch 58/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4599 - loss:
1.5182 - val_accuracy: 0.4192 - val_loss: 1.6974
Epoch 59/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4520 - loss:
1.5188 - val_accuracy: 0.4221 - val_loss: 1.6893
Epoch 60/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4622 - loss:
1.5099 - val_accuracy: 0.4369 - val_loss: 1.6524
Epoch 61/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4604 - loss:
```

```
1.5110 - val_accuracy: 0.4275 - val_loss: 1.6563
Epoch 62/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4594 - loss:
1.5101 - val_accuracy: 0.4104 - val_loss: 1.7132
Epoch 63/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4626 - loss:
1.4984 - val_accuracy: 0.4307 - val_loss: 1.6487
Epoch 64/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.4554 - loss:
1.5143 - val_accuracy: 0.4296 - val_loss: 1.6843
Epoch 65/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4575 - loss:
1.4995 - val_accuracy: 0.4388 - val_loss: 1.6229
Epoch 66/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4617 - loss:
1.4978 - val_accuracy: 0.4152 - val_loss: 1.6846
Epoch 67/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4613 - loss:
1.4999 - val_accuracy: 0.4422 - val_loss: 1.6246
Epoch 68/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4649 - loss:
1.4912 - val_accuracy: 0.4410 - val_loss: 1.6326
Epoch 69/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4665 - loss:
1.4850 - val_accuracy: 0.4287 - val_loss: 1.6616
Epoch 70/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4673 - loss:
1.4810 - val_accuracy: 0.4217 - val_loss: 1.6715
Epoch 71/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4636 - loss:
1.5011 - val_accuracy: 0.4268 - val_loss: 1.6793
Epoch 72/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4716 - loss:
1.4852 - val_accuracy: 0.4218 - val_loss: 1.6796
Epoch 73/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4667 - loss:
1.4862 - val_accuracy: 0.4233 - val_loss: 1.6713
Epoch 74/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4663 - loss:
1.4919 - val_accuracy: 0.4298 - val_loss: 1.6581
Epoch 75/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4666 - loss:
1.4805 - val_accuracy: 0.4419 - val_loss: 1.6339
Epoch 76/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4649 - loss:
1.4863 - val_accuracy: 0.4311 - val_loss: 1.6676
Epoch 77/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4721 - loss:
1.4741 - val_accuracy: 0.4374 - val_loss: 1.6367
```

```
Epoch 78/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4700 - loss:
1.4750 - val_accuracy: 0.4512 - val_loss: 1.6107
Epoch 79/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4763 - loss:
1.4669 - val_accuracy: 0.4254 - val_loss: 1.6461
Epoch 80/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4726 - loss:
1.4722 - val_accuracy: 0.4372 - val_loss: 1.6344
Epoch 81/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4766 - loss:
1.4641 - val_accuracy: 0.4181 - val_loss: 1.7106
Epoch 82/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4811 - loss:
1.4620 - val_accuracy: 0.4402 - val_loss: 1.6641
Epoch 83/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4741 - loss:
1.4752 - val_accuracy: 0.4422 - val_loss: 1.6487
Epoch 84/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.4730 - loss:
1.4726 - val_accuracy: 0.4320 - val_loss: 1.6590
Epoch 85/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4782 - loss:
1.4625 - val_accuracy: 0.4227 - val_loss: 1.6967
Epoch 86/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4793 - loss:
1.4529 - val_accuracy: 0.4266 - val_loss: 1.6806
Epoch 87/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4735 - loss:
1.4723 - val_accuracy: 0.4209 - val_loss: 1.6828
Epoch 88/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4695 - loss:
1.4720 - val_accuracy: 0.4230 - val_loss: 1.6673
Epoch 89/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4767 - loss:
1.4644 - val_accuracy: 0.4066 - val_loss: 1.7122
Epoch 90/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4789 - loss:
1.4591 - val_accuracy: 0.4291 - val_loss: 1.6721
Epoch 91/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4775 - loss:
1.4636 - val_accuracy: 0.4484 - val_loss: 1.6095
Epoch 92/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4775 - loss:
1.4596 - val_accuracy: 0.4294 - val_loss: 1.6694
Epoch 93/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4859 - loss:
1.4537 - val_accuracy: 0.4363 - val_loss: 1.6515
Epoch 94/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4773 - loss:
1.4628 - val_accuracy: 0.4334 - val_loss: 1.6674
Epoch 95/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4833 - loss:
1.4429 - val_accuracy: 0.4326 - val_loss: 1.6519
Epoch 96/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4780 - loss:
1.4667 - val_accuracy: 0.4304 - val_loss: 1.6615
Epoch 97/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4781 - loss:
1.4589 - val_accuracy: 0.4227 - val_loss: 1.6756
Epoch 98/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4846 - loss:
1.4494 - val_accuracy: 0.4451 - val_loss: 1.6357
Epoch 99/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4816 - loss:
1.4415 - val_accuracy: 0.4355 - val_loss: 1.6532
Epoch 100/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4816 - loss:
1.4477 - val_accuracy: 0.4220 - val_loss: 1.6958
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.4659 - loss:
1.5944
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.4206 - loss:
1.6903
Modelo v2 - Acurácia no teste: 0.4220
```

#3. Retirar as camadas de dropout e aplicar regularização L2 Novo modelo com regularização L2:

```
from tensorflow.keras.regularizers import l2

def create_residual_model_v3(n1, n2, n3, num_classes, input_shape=(32,
32, 3)):
    inputs = layers.Input(shape=input_shape)
    x = layers.Flatten()(inputs)

    # Bloco 1 com L2 regularization
    x = layers.Dense(n1, activation='relu',
kernel_regularizer=l2(0.01))(x)
    x = residual_block(x, n1)

    # Bloco 2 com L2 regularization
    x = layers.Dense(n2, activation='relu',
kernel_regularizer=l2(0.01))(x)
    x = residual_block(x, n2)

    # Bloco 3 com L2 regularization
    x = layers.Dense(n3, activation='relu',
kernel_regularizer=l2(0.01))(x)
```

```python
    x = residual_block(x, n3)

    # Camada de saída
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = models.Model(inputs=inputs, outputs=outputs)
    return model

# Criar e compilar o modelo com L2 regularization
rna_v3 = create_residual_model_v3(n1, n2, n3, num_classes)
rna_v3.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Treinar o modelo
history_v3 = rna_v3.fit(x_train_norm, y_train_hot, epochs=100,
batch_size=256, validation_data=(x_test_norm, y_test_hot))

# Avaliar o modelo
train_loss_v3, train_accuracy_v3 = rna_v3.evaluate(x_train_norm,
y_train_hot)
test_loss_v3, test_accuracy_v3 = rna_v3.evaluate(x_test_norm,
y_test_hot)

print(f"Modelo v3 - Acurácia no teste: {test_accuracy_v3:.4f}")
```

```
Epoch 1/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.2099 - loss:
9.0983 - val_accuracy: 0.3245 - val_loss: 2.8089
Epoch 2/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.3386 - loss:
2.5926 - val_accuracy: 0.3095 - val_loss: 2.3499
Epoch 3/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3522 - loss:
2.1701 - val_accuracy: 0.3773 - val_loss: 2.0278
Epoch 4/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3723 - loss:
1.9915 - val_accuracy: 0.3347 - val_loss: 2.0090
Epoch 5/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.3738 - loss:
1.9180 - val_accuracy: 0.3520 - val_loss: 1.9537
Epoch 6/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.3824 - loss:
1.8602 - val_accuracy: 0.4001 - val_loss: 1.8025
Epoch 7/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.3897 - loss:
1.8200 - val_accuracy: 0.3964 - val_loss: 1.7805
Epoch 8/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3995 - loss:
1.7748 - val_accuracy: 0.3548 - val_loss: 1.8937
Epoch 9/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4048 - loss:
1.7693 - val_accuracy: 0.4127 - val_loss: 1.7674
Epoch 10/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4139 - loss:
1.7303 - val_accuracy: 0.4155 - val_loss: 1.7323
Epoch 11/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4141 - loss:
1.7227 - val_accuracy: 0.3868 - val_loss: 1.8086
Epoch 12/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4209 - loss:
1.7100 - val_accuracy: 0.3839 - val_loss: 1.8354
Epoch 13/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4252 - loss:
1.7042 - val_accuracy: 0.4347 - val_loss: 1.6765
Epoch 14/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4282 - loss:
1.6909 - val_accuracy: 0.4139 - val_loss: 1.7482
Epoch 15/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4331 - loss:
1.6661 - val_accuracy: 0.4230 - val_loss: 1.7399
Epoch 16/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4264 - loss:
1.6940 - val_accuracy: 0.4423 - val_loss: 1.6623
Epoch 17/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4372 - loss:
1.6587 - val_accuracy: 0.4340 - val_loss: 1.7034
Epoch 18/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4350 - loss:
1.6726 - val_accuracy: 0.4434 - val_loss: 1.6606
Epoch 19/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4325 - loss:
1.6741 - val_accuracy: 0.4571 - val_loss: 1.6191
Epoch 20/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4518 - loss:
1.6197 - val_accuracy: 0.4345 - val_loss: 1.6831
Epoch 21/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4560 - loss:
1.6113 - val_accuracy: 0.4276 - val_loss: 1.7024
Epoch 22/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4431 - loss:
1.6351 - val_accuracy: 0.4634 - val_loss: 1.5991
Epoch 23/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4604 - loss:
1.6057 - val_accuracy: 0.4469 - val_loss: 1.6435
Epoch 24/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4603 - loss:
1.6073 - val_accuracy: 0.4237 - val_loss: 1.6949
Epoch 25/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4648 - loss:
```

```
1.5953 - val_accuracy: 0.4499 - val_loss: 1.6615
Epoch 26/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4548 - loss:
1.6129 - val_accuracy: 0.4720 - val_loss: 1.5930
Epoch 27/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4754 - loss:
1.5700 - val_accuracy: 0.4135 - val_loss: 1.7559
Epoch 28/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4690 - loss:
1.5817 - val_accuracy: 0.4756 - val_loss: 1.5992
Epoch 29/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4708 - loss:
1.5820 - val_accuracy: 0.4587 - val_loss: 1.6161
Epoch 30/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4699 - loss:
1.5737 - val_accuracy: 0.4284 - val_loss: 1.6921
Epoch 31/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4720 - loss:
1.5782 - val_accuracy: 0.4658 - val_loss: 1.6084
Epoch 32/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4780 - loss:
1.5536 - val_accuracy: 0.4573 - val_loss: 1.6216
Epoch 33/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4884 - loss:
1.5293 - val_accuracy: 0.4355 - val_loss: 1.6791
Epoch 34/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4863 - loss:
1.5446 - val_accuracy: 0.4722 - val_loss: 1.5863
Epoch 35/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4862 - loss:
1.5425 - val_accuracy: 0.4550 - val_loss: 1.6451
Epoch 36/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4806 - loss:
1.5573 - val_accuracy: 0.4563 - val_loss: 1.6381
Epoch 37/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4911 - loss:
1.5272 - val_accuracy: 0.4903 - val_loss: 1.5476
Epoch 38/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4974 - loss:
1.5086 - val_accuracy: 0.4511 - val_loss: 1.6478
Epoch 39/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.4900 - loss:
1.5344 - val_accuracy: 0.4756 - val_loss: 1.6152
Epoch 40/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.4930 - loss:
1.5158 - val_accuracy: 0.4740 - val_loss: 1.5787
Epoch 41/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4932 - loss:
1.5200 - val_accuracy: 0.4536 - val_loss: 1.6620
```

```
Epoch 42/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4964 - loss:
1.5126 - val_accuracy: 0.4521 - val_loss: 1.6405
Epoch 43/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.4876 - loss:
1.5317 - val_accuracy: 0.4684 - val_loss: 1.5997
Epoch 44/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5004 - loss:
1.5073 - val_accuracy: 0.4823 - val_loss: 1.5818
Epoch 45/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5112 - loss:
1.4844 - val_accuracy: 0.4733 - val_loss: 1.5910
Epoch 46/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5082 - loss:
1.4975 - val_accuracy: 0.4450 - val_loss: 1.6689
Epoch 47/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5064 - loss:
1.4894 - val_accuracy: 0.4618 - val_loss: 1.6250
Epoch 48/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5134 - loss:
1.4715 - val_accuracy: 0.4650 - val_loss: 1.6349
Epoch 49/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5084 - loss:
1.4774 - val_accuracy: 0.4739 - val_loss: 1.5759
Epoch 50/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.5167 - loss:
1.4617 - val_accuracy: 0.4806 - val_loss: 1.5982
Epoch 51/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5209 - loss:
1.4498 - val_accuracy: 0.4795 - val_loss: 1.5785
Epoch 52/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5167 - loss:
1.4638 - val_accuracy: 0.4836 - val_loss: 1.5756
Epoch 53/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5304 - loss:
1.4325 - val_accuracy: 0.4837 - val_loss: 1.5869
Epoch 54/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5193 - loss:
1.4655 - val_accuracy: 0.4848 - val_loss: 1.5741
Epoch 55/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5357 - loss:
1.4223 - val_accuracy: 0.4655 - val_loss: 1.6136
Epoch 56/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5250 - loss:
1.4375 - val_accuracy: 0.4852 - val_loss: 1.5686
Epoch 57/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.5229 - loss:
1.4451 - val_accuracy: 0.4763 - val_loss: 1.5927
Epoch 58/100
```

```
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5333 - loss:
1.4312 - val_accuracy: 0.4794 - val_loss: 1.5760
Epoch 59/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5331 - loss:
1.4297 - val_accuracy: 0.4893 - val_loss: 1.5762
Epoch 60/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.5340 - loss:
1.4226 - val_accuracy: 0.4598 - val_loss: 1.6487
Epoch 61/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.5270 - loss:
1.4397 - val_accuracy: 0.5021 - val_loss: 1.5385
Epoch 62/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.5461 - loss:
1.3986 - val_accuracy: 0.4802 - val_loss: 1.5833
Epoch 63/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5444 - loss:
1.3892 - val_accuracy: 0.4780 - val_loss: 1.6140
Epoch 64/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5458 - loss:
1.4036 - val_accuracy: 0.4839 - val_loss: 1.6020
Epoch 65/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5446 - loss:
1.3933 - val_accuracy: 0.4811 - val_loss: 1.6152
Epoch 66/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5494 - loss:
1.3806 - val_accuracy: 0.4597 - val_loss: 1.6651
Epoch 67/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5510 - loss:
1.3862 - val_accuracy: 0.4930 - val_loss: 1.5701
Epoch 68/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5550 - loss:
1.3721 - val_accuracy: 0.4629 - val_loss: 1.6763
Epoch 69/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5573 - loss:
1.3678 - val_accuracy: 0.4711 - val_loss: 1.6192
Epoch 70/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5634 - loss:
1.3440 - val_accuracy: 0.4756 - val_loss: 1.6458
Epoch 71/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5490 - loss:
1.3935 - val_accuracy: 0.4939 - val_loss: 1.5978
Epoch 72/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5663 - loss:
1.3493 - val_accuracy: 0.4912 - val_loss: 1.5837
Epoch 73/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5678 - loss:
1.3531 - val_accuracy: 0.4734 - val_loss: 1.6578
Epoch 74/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5621 - loss:
```

```
1.3531 - val_accuracy: 0.4653 - val_loss: 1.6469
Epoch 75/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5685 - loss:
1.3434 - val_accuracy: 0.4889 - val_loss: 1.6033
Epoch 76/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5708 - loss:
1.3338 - val_accuracy: 0.4841 - val_loss: 1.6139
Epoch 77/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5780 - loss:
1.3220 - val_accuracy: 0.4730 - val_loss: 1.6764
Epoch 78/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5633 - loss:
1.3535 - val_accuracy: 0.4799 - val_loss: 1.6599
Epoch 79/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5729 - loss:
1.3295 - val_accuracy: 0.4868 - val_loss: 1.6715
Epoch 80/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5764 - loss:
1.3217 - val_accuracy: 0.4849 - val_loss: 1.6363
Epoch 81/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.5763 - loss:
1.3304 - val_accuracy: 0.4863 - val_loss: 1.6168
Epoch 82/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.5826 - loss:
1.3087 - val_accuracy: 0.4922 - val_loss: 1.6393
Epoch 83/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5821 - loss:
1.3057 - val_accuracy: 0.4908 - val_loss: 1.6467
Epoch 84/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5861 - loss:
1.3000 - val_accuracy: 0.4842 - val_loss: 1.6810
Epoch 85/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5834 - loss:
1.3052 - val_accuracy: 0.4675 - val_loss: 1.6995
Epoch 86/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5839 - loss:
1.3097 - val_accuracy: 0.4788 - val_loss: 1.6656
Epoch 87/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5927 - loss:
1.2914 - val_accuracy: 0.4904 - val_loss: 1.6565
Epoch 88/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5976 - loss:
1.2725 - val_accuracy: 0.4660 - val_loss: 1.7698
Epoch 89/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5900 - loss:
1.3071 - val_accuracy: 0.4891 - val_loss: 1.7567
Epoch 90/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6066 - loss:
1.3193 - val_accuracy: 0.4823 - val_loss: 1.6823
```

```
Epoch 91/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.6062 - loss:
1.2638 - val_accuracy: 0.4883 - val_loss: 1.6733
Epoch 92/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6002 - loss:
1.2774 - val_accuracy: 0.4847 - val_loss: 1.6650
Epoch 93/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.5970 - loss:
1.2765 - val_accuracy: 0.4730 - val_loss: 1.7226
Epoch 94/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6096 - loss:
1.2461 - val_accuracy: 0.4921 - val_loss: 1.6389
Epoch 95/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6121 - loss:
1.2353 - val_accuracy: 0.4676 - val_loss: 1.7281
Epoch 96/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6026 - loss:
1.2710 - val_accuracy: 0.4920 - val_loss: 1.6649
Epoch 97/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6109 - loss:
1.2327 - val_accuracy: 0.4913 - val_loss: 1.6927
Epoch 98/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6195 - loss:
1.2198 - val_accuracy: 0.4749 - val_loss: 1.7341
Epoch 99/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6139 - loss:
1.2331 - val_accuracy: 0.4848 - val_loss: 1.7196
Epoch 100/100
196/196 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.6142 - loss:
1.2320 - val_accuracy: 0.4887 - val_loss: 1.7125
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.6229 - loss:
1.2104
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.4921 - loss:
1.6971
Modelo v3 - Acurácia no teste: 0.4887
```

#Conclusão O Modelo v1 foi o mais bem-sucedido devido ao aumento da capacidade de aprendizado com blocos residuais adicionais e mais unidades densas. A introdução de regularizações (v2 e v3) foi menos eficaz para este caso, pois não havia sinais claros de overfitting no modelo base.