Visualização por Computador e Processamento de Imagem

Diogo Vieira PG50518
Laura Rodrigues PG50542
Mariana Marques PG50633

Preparação do *Dataset*

- Balanceamento de Dados

```
classes = os.listdir(f'{data_path}/train_balanced_{IMAGES_PER_CLASS}')
list_img = []
for cla in classes:
    list_img = os.listdir(f'{data_path}/train_balanced_{IMAGES_PER_CLASS}/{cla}')
    random.shuffle(list_img)
    for k in range(len(list_img), IMAGES_PER_CLASS):
        filename = f'{data_path}/train_balanced_{IMAGES_PER_CLASS}/{cla}/{list_img[(k - len(list_img)) % len(list_img)]}'
        if not filename.endswith('png'):
          continue
        im = Image.open(filename)
        r = random.uniform(-10.0,10.0)
        im = im.rotate(r)
        r1 = random.uniform(-3.0,3.0)
        r2 = random.uniform(-3.0,3.0)
        im = im.transform(im.size, Image.Transform.AFFINE, (1, 0, r1, 0, 1, r2))
        r = random.uniform(1.0, 1.3)
        im = ImageEnhance.Sharpness(im.convert('RGB'))
        im = im.enhance(r)
        r = random.uniform(1.0, 1.3)
        im = ImageEnhance.Contrast(im)
        im = im.enhance(r)
        im = im.resize((32,32))
        im.save(f'{data_path}/train_balanced_{IMAGES_PER_CLASS}/{cla}/_{k}.png')
```

Data Augmentation

Data Augmentation + Aplicação de Filtros

1. *Cor*
- **Brightness**
- **Contrast**
- **Hue**
- **Saturation**

2. *Transformações Geométricas*
- **Rotate**
- **Shear**
- **Translate**
- **Crop**

3. *Contornos*
- **Edges**

Cor

```
def process_brightness(image, label):

    img = tf.clip_by_value(tfa.image.random_hsv_in_yiq(image, 0.0, 1.0, 1.0, 0.1, 3.0),0,1)
    return img, label
```

```
def process_saturation(image, label):

    img = tf.clip_by_value(tfa.image.random_hsv_in_yiq(image, 0.0, 1.0, 3.0, 1.0, 1.0),0,1)
    return img, label
```

```
def process_contrast(image, label):

    img = tf.clip_by_value(tf.image.random_contrast(image, lower=0.1, upper=3.0, seed=None), 0, 1)
    return img, label
```

```
def process_hue(image, label):

    img = tf.image.random_hue(image, max_delta=0.2, seed=None)
    return img, label
```

Transformações Geométricas

```python
def process_rotate(image, label):

    img = tfa.image.rotate(image, tf.random.uniform(shape=(), minval=-0.175, maxval=0.175))
    return img, label
```

```python
def process_shear(image, label):

    img = tfa.image.rotate(image, tf.random.uniform(shape=(), minval=-0.175, maxval=0.175))
    sx = tf.random.uniform(shape=(), minval=-0.1, maxval=0.1, dtype=tf.dtypes.float32)
    img = tfa.image.transform(img, [1, sx, -sx*32,   0,1,0,  0,0])
    return img, label
```

```python
def process_translate(image, label):

    img = tfa.image.rotate(image, tf.random.uniform(shape=(), minval=-0.175, maxval=0.175))
    tx = tf.random.uniform(shape=(), minval=-3, maxval=3, dtype=tf.dtypes.float32)
    ty = tf.random.uniform(shape=(), minval=-3, maxval=3, dtype=tf.dtypes.float32)
    img = tfa.image.translate(img, [tx,ty])
    return img, label
```

```python
def process_crop(image, label):

    c = tf.random.uniform(shape=(), minval=24, maxval=32, dtype=tf.dtypes.float32)
    img = tf.image.random_crop(image, size=[c,c,3])
    img = tf.image.resize(img ,size= [32,32])
    return img, label
```

Contornos

```python
def process_edges(image, label):
    # Expand dimensions to match expected rank
    expanded_image = tf.expand_dims(image, axis=0)
    # Apply Sobel filter to the expanded image
    sobel = tf.image.sobel_edges(expanded_image)
    # Compute gradient magnitude
    gradient_magnitude = tf.sqrt(tf.square(sobel[..., 0]) + tf.square(sobel[..., 1]))
    # Clip values between 0 and 1
    clipped_img = tf.clip_by_value(gradient_magnitude / 8, 0, 1)
    # Remove the extra dimension
    img = tf.squeeze(clipped_img, axis=0)
    return img, label
```

Treinar os Modelos

- CNN

```
def model_III(classCount, imgSize, channels):

    model = Sequential()

    model.add(Conv2D(128, (5, 5),
                     input_shape=(imgSize, imgSize, channels)
                     ))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.01))

    model.add(Conv2D(128, (5, 5) ))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.01))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(256, (5, 5) ) )
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.01))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(128))
    model.add(LeakyReLU(alpha=0.01))
    model.add(Dropout(0.2))

    model.add(Dense(classCount, activation='softmax'))


    opt = Adam(lr=0.0001)
    model.compile(optimizer = opt, loss='categorical_crossentropy', metrics=[ 'accuracy'])
    return model
```

Modelos Criados

- 1 para cada filtro
  - Brightness
  - Contrast
  - Hue
  - Saturation
  - Rotate
  - Translate
  - Shear
  - Crop
  - Sobel
- 3 para grupos de filtros
  - Cor
  - Transformações geométricas
  - Todos

Resultados

Original

Modelo	loss	accuracy
No filters	0.1648	0.9839
Crop	0.1796	0.9739
Contrast	0.1376	0.9864
Brightness	0.2021	0.9804
Hue	0.1949	0.9763
Translate	0.1059	0.9813
Rotate	0.1411	0.9793
Saturation	0.1666	0.9804
Shear	0.0947	0.9850
Sobel	0.2133	0.9780
Colors_Concat	0.2700	0.9836
Geo_Concat	0.1893	0.9726
All	0.0755	0.9908

Balanceado

Modelo	loss	accuracy
No filters	0.0931	0.9836
Crop	0.0990	0.9807
Contrast	0.0925	0.9868
Brightness	0.1632	0.9820
Hue	0.0781	0.9874
Translate	0.1064	0.9838
Rotate	0.0799	0.9875
Saturation	0.0802	0.9859
Shear	0.0727	0.9866
Sobel	0.0979	0.9884
Colors_Concat	0.1181	0.9876
Geo_Concat	0.0822	0.9909
All	0.1138	0.9892

Ensemble

Voting Ensemble

2 grupos:

- 9 modelos (1 para cada filtro)

num_images	all_correct	all_incorrect	maj_vote	tie	maj_wrong	log_ok	log_ko
12630	12023	4	523	20	60	12561	69

A accuracy total é 0.9945

A accuracy média deste modelo é de 98.55. Sendo a pior, 98.07, e a melhor 98.84.

- 3 modelos
    - Grupo Cores
    - Grupo Transformações geométricas
    - Sobel

num_images	all_correct	all_incorrect	maj_vote	tie	maj_wrong	log_ok	log_ko
12630	12330	11	209	41	39	12573	57

A accuracy total é 0.9955

A accuracy média deste modelo é de 98.894. Sendo a pior, 98.76, e a melhor 99.09.

Conclusões e Trabalho Futuro

Gostaríamos de ter conseguido concretizar as implementações de Stacking Ensemble e dos filtros de Fourier, Gray Scale e Perlin Noise. Ficam assim estes tópicos para trabalho futuro de modo a tentar obter melhores resultados.