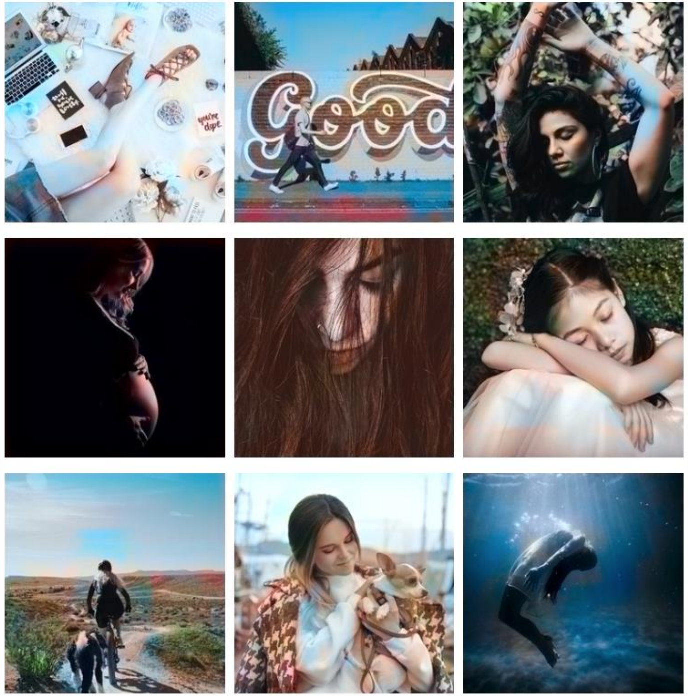
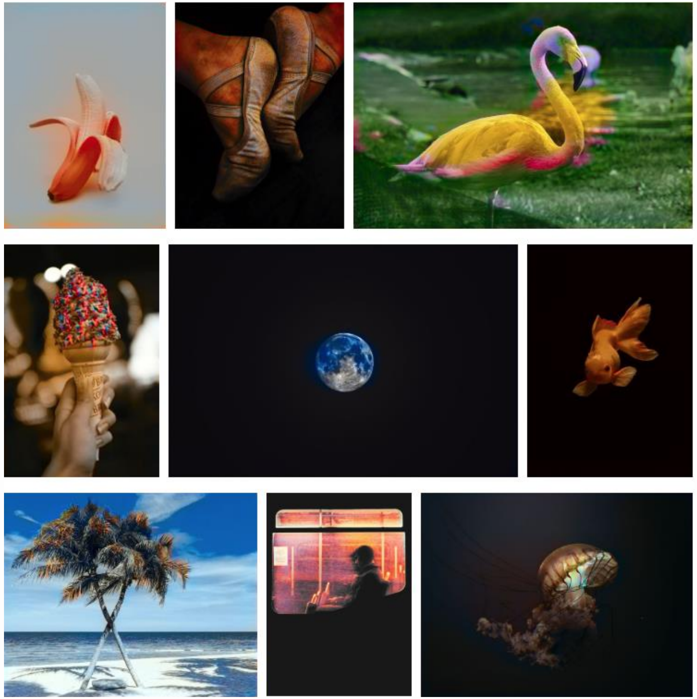
Colouring grayscale photos

Xarxes Neuronals i Aprenentatge Profund

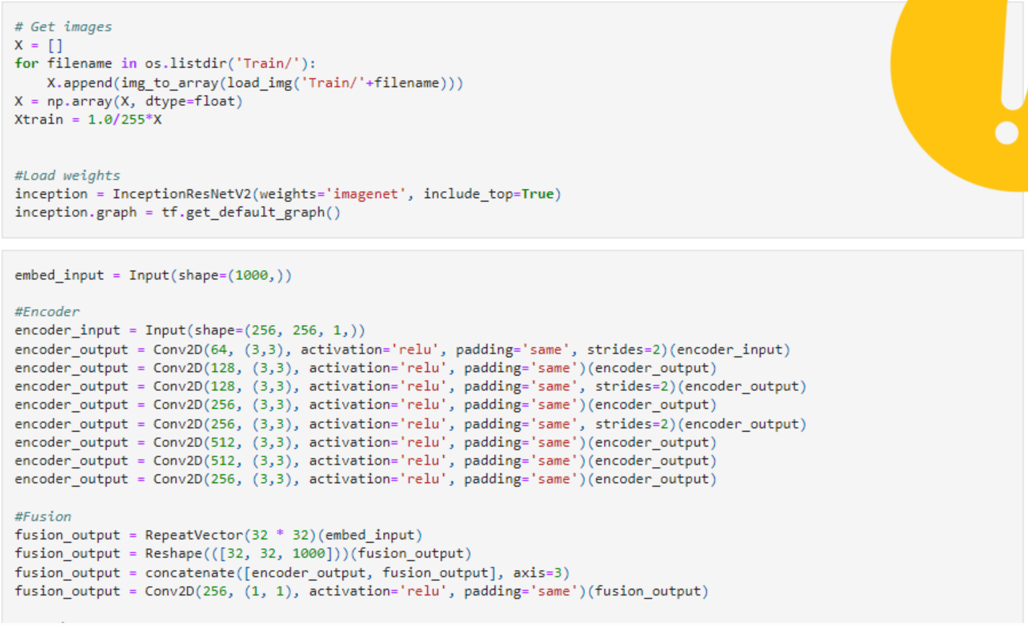
Grup 1

Introducción





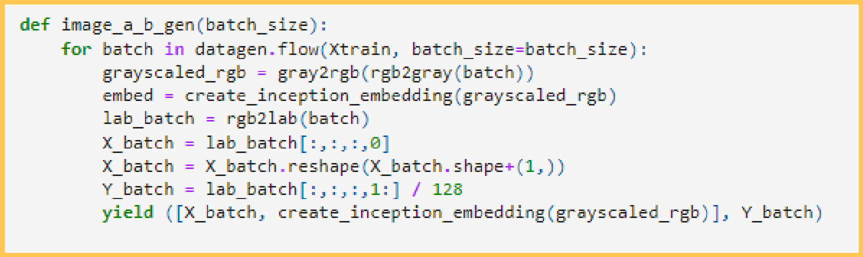
Punto de inicio del proyecto



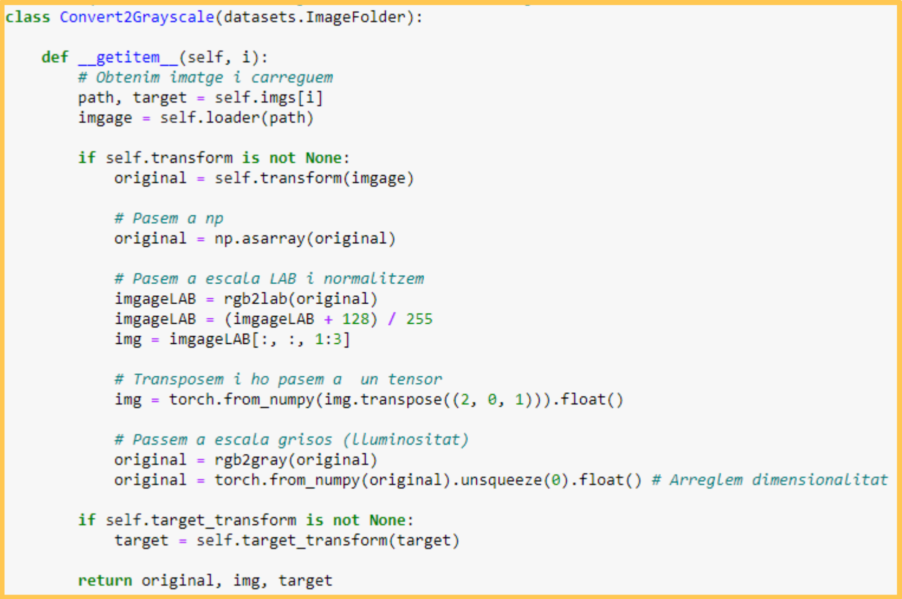
```
# Get images
X = []
for filename in os.listdir('Train/'):
    X.append(img_to_array(load_img('Train/'+filename)))
X = np.array(X, dtype=float)
Xtrain = 1.0/255*X


#Load weights
inception = InceptionResNetV2(weights='imagenet', include_top=True)
inception.graph = tf.get_default_graph()
```

```
embed_input = Input(shape=(1000,))

#Encoder
encoder_input = Input(shape=(256, 256, 1,))
encoder_output = Conv2D(64, (3,3), activation='relu', padding='same', strides=2)(encoder_input)
encoder_output = Conv2D(128, (3,3), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(128, (3,3), activation='relu', padding='same', strides=2)(encoder_output)
encoder_output = Conv2D(256, (3,3), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(256, (3,3), activation='relu', padding='same', strides=2)(encoder_output)
encoder_output = Conv2D(512, (3,3), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(512, (3,3), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(256, (3,3), activation='relu', padding='same')(encoder_output)

#Fusion
fusion_output = RepeatVector(32 * 32)(embed_input)
fusion_output = Reshape(([32, 32, 1000]))(fusion_output)
fusion_output = concatenate([encoder_output, fusion_output], axis=3)
fusion_output = Conv2D(256, (1, 1), activation='relu', padding='same')(fusion_output)
```

Punto de inicio del proyecto

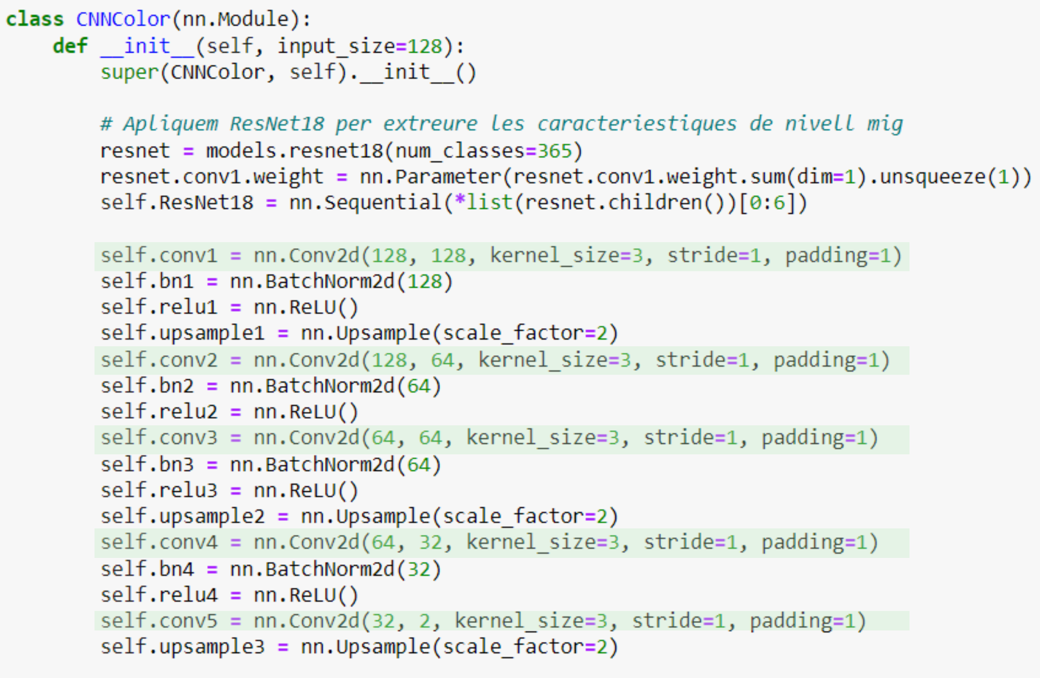


```python
def image_a_b_gen(batch_size):
    for batch in datagen.flow(Xtrain, batch_size=batch_size):
        grayscaled_rgb = gray2rgb(rgb2gray(batch))
        embed = create_inception_embedding(grayscaled_rgb)
        lab_batch = rgb2lab(batch)
        X_batch = lab_batch[:,:,:,0]
        X_batch = X_batch.reshape(X_batch.shape+(1,))
        Y_batch = lab_batch[:,:,:,1:] / 128
        yield ([X_batch, create_inception_embedding(grayscaled_rgb)], Y_batch)
```



```python
class Convert2Grayscale(datasets.ImageFolder):

    def __getitem__(self, i):
        # Obtenim imatge i carreguem
        path, target = self.imgs[i]
        imgage = self.loader(path)

        if self.transform is not None:
            original = self.transform(imgage)

            # Pasem a np
            original = np.asarray(original)

            # Pasem a escala LAB i normalitzem
            imgageLAB = rgb2lab(original)
            imgageLAB = (imgageLAB + 128) / 255
            img = imgageLAB[:, :, 1:3]

            # Transposem i ho pasem a  un tensor
            img = torch.from_numpy(img.transpose((2, 0, 1))).float()

            # Passem a escala grisos (lluminositat)
            original = rgb2gray(original)
            original = torch.from_numpy(original).unsqueeze(0).float() # Arreglem dimensionalitat

        if self.target_transform is not None:
            target = self.target_transform(target)

        return original, img, target
```

Modelo



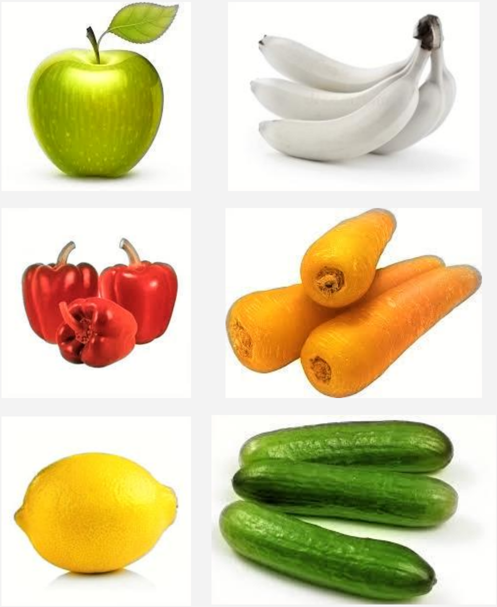
```python
class CNNColor(nn.Module):
    def __init__(self, input_size=128):
        super(CNNColor, self).__init__()

        # Apliquem ResNet18 per extreure les caracteriestiques de nivell mig
        resnet = models.resnet18(num_classes=365)
        resnet.conv1.weight = nn.Parameter(resnet.conv1.weight.sum(dim=1).unsqueeze(1))
        self.ResNet18 = nn.Sequential(*list(resnet.children())[0:6])

        self.conv1 = nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(128)
        self.relu1 = nn.ReLU()
        self.upsample1 = nn.Upsample(scale_factor=2)
        self.conv2 = nn.Conv2d(128, 64, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(64)
        self.relu2 = nn.ReLU()
        self.conv3 = nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1)
        self.bn3 = nn.BatchNorm2d(64)
        self.relu3 = nn.ReLU()
        self.upsample2 = nn.Upsample(scale_factor=2)
        self.conv4 = nn.Conv2d(64, 32, kernel_size=3, stride=1, padding=1)
        self.bn4 = nn.BatchNorm2d(32)
        self.relu4 = nn.ReLU()
        self.conv5 = nn.Conv2d(32, 2, kernel_size=3, stride=1, padding=1)
        self.upsample3 = nn.Upsample(scale_factor=2)
```

Datasets y entrenamiento

Train = 80%
Validate = 20%

Faces +750
15 epochs | ≈15-20 min

Food +685
15 epochs | ≈ 15-20 min



Random +31.784
20 epochs | ≈ 7 h

Experimentos

Learning Rate Schedule

- Rendimiento
- Estabilidad

Entrenar el modelo con otros datsets

- Food dataset
- Random dataset

Probar el modelo con otras imagenes

Experimentos pendientes

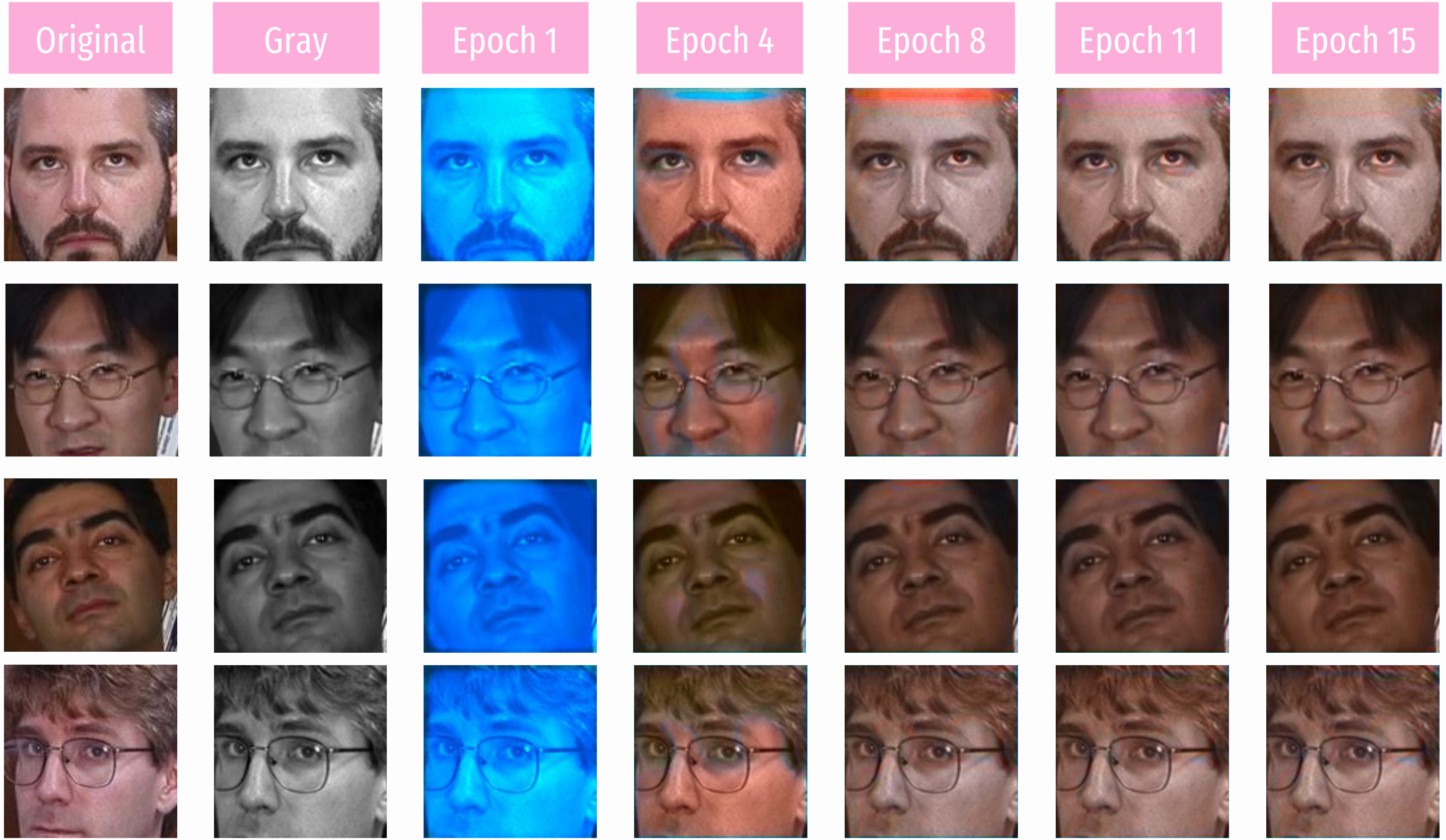
Ejecutar con más epochs

Probar otros optimizadores/ criterios

Probar otros modelos

FACES DATASET

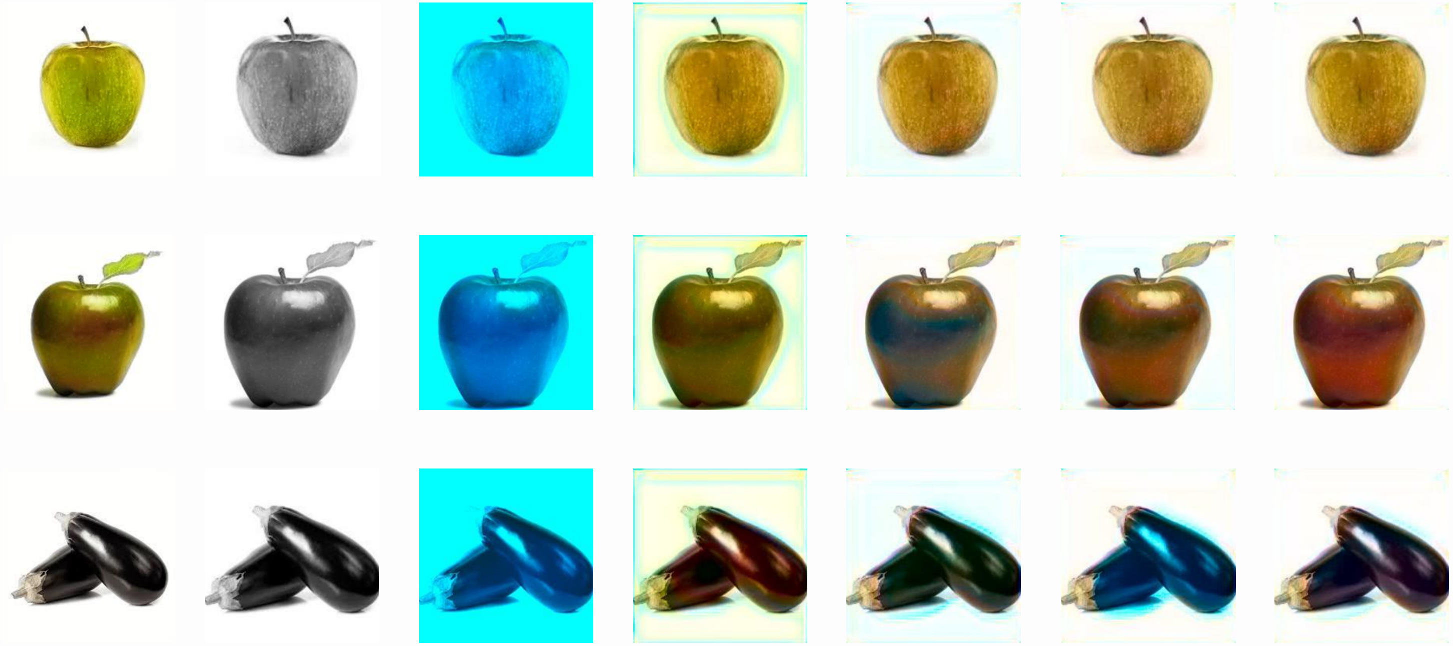
Resultados



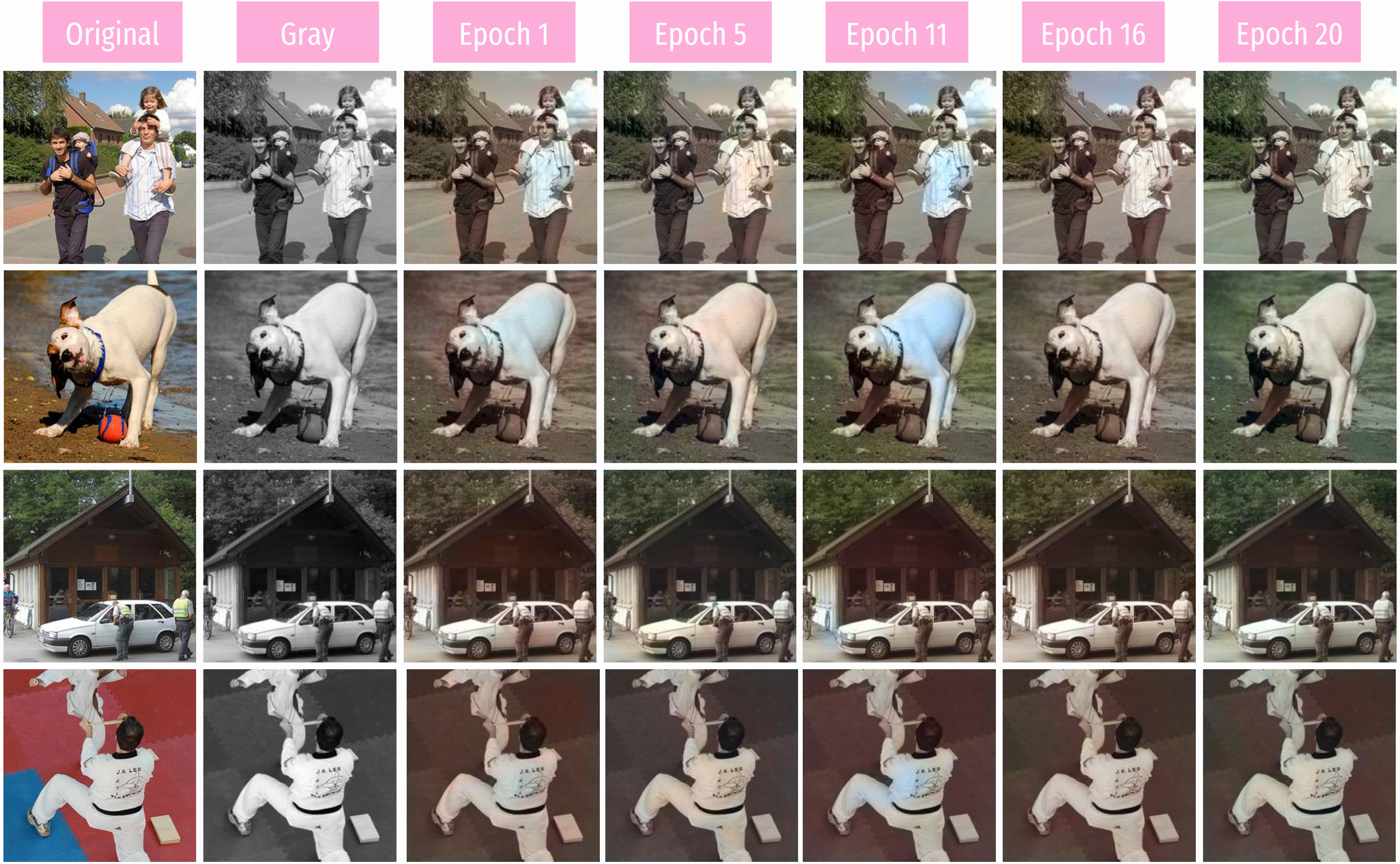
FOOD DATASET

Resultados

Original | Gray | Epoch 1 | Epoch 4 | Epoch 8 | Epoch 11 | Epoch 15



RANDOM DATASET



Conclusiones

Faces

- Colorea de manera correcta caras
- No da mucha información del funcionamiento del modelo

Food

- Colorea de manera correcta frutas y verduras
- No se puede generalizar para otros datos

Random

- Con un dataset tan malo el modelo no aprende bien a colorear

Learning Rate Schedule

- Rendimiento
- Estabilidad