UNIVERSIDAD NACIONAL DE INGENIERÍA

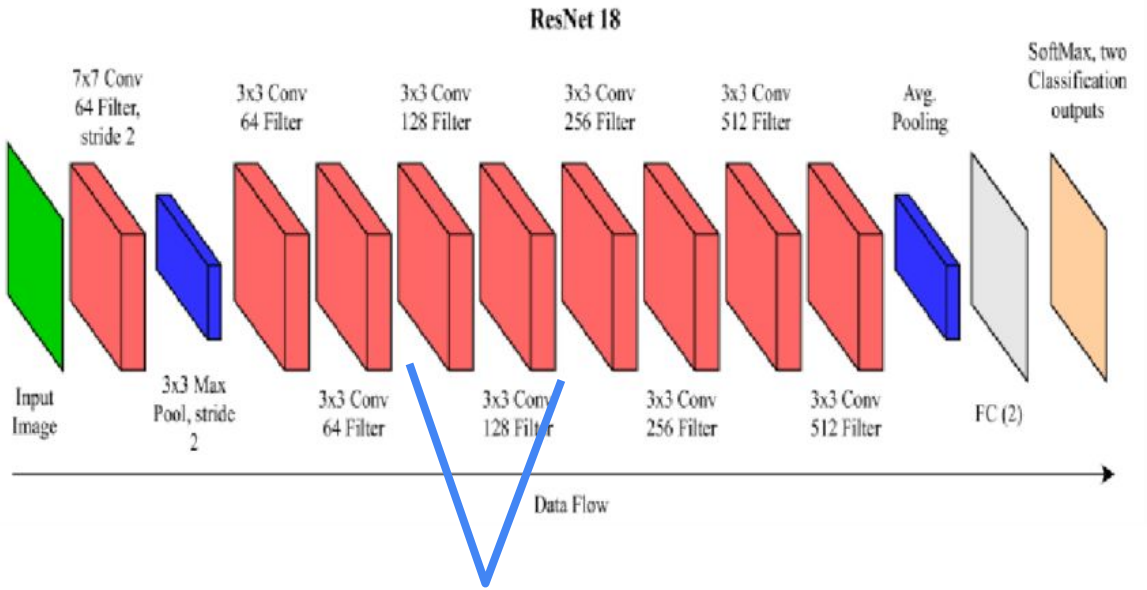
REDES NEURONALES Y APRENDIZAJE PROFUNDO

Tarea 1

Integrantes

- Alejos Yarasca Fiorella Andrea (fiorella.alejos.y@uni.pe)
- Llana Chavez Walter Rodolfo (walter.llana.c@uni.pe)
- Luna Jaramillo Juan Marcos (juan.luna.j@uni.pe)
- Medina Rodríguez Henry (hmedinar@uni.pe)
- Salazar Vega Edwin Martín (edwin@iartificial.io)

ResNet-18



Bloque residual

Incluye:

- Capas iniciales:
  - Convolución (kernel 7x7)
  - Max Pooling (3x3)
- 4 Grupos residuales:
  - Grupo 1: 64 filtros
  - Grupo 2: 128 filtros
  - Grupo 3: 256 filtros
  - Grupo 4: 512 filtros
- Capas finales:
  - Average pooling (7x7)
  - Fully connected

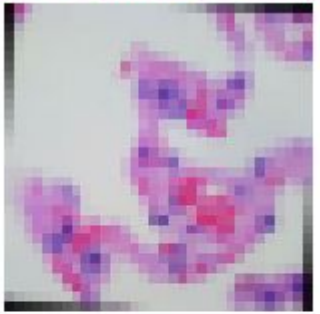
ResNet-18

Testing

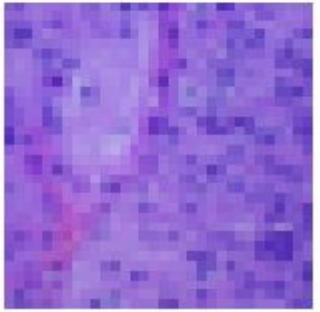
```
[ ]  # Testing the model
     model.eval()  # Set the model to evaluation mode
     correct = 0
     total = 0
     with torch.no_grad():
         for data in val_loader:
             images, labels = data[0].to(device), data[1].to(device)
             outputs = model(images)
             _, predicted = torch.max(outputs, 1)
             total += labels.size(0)
             correct += (predicted == labels).sum().item()

     print(f'Accuracy of the model on the 10,000 test images: {100 * correct / total:.2f}%')
```

```
Accuracy of the model on the 10,000 test images: 98.20%
```

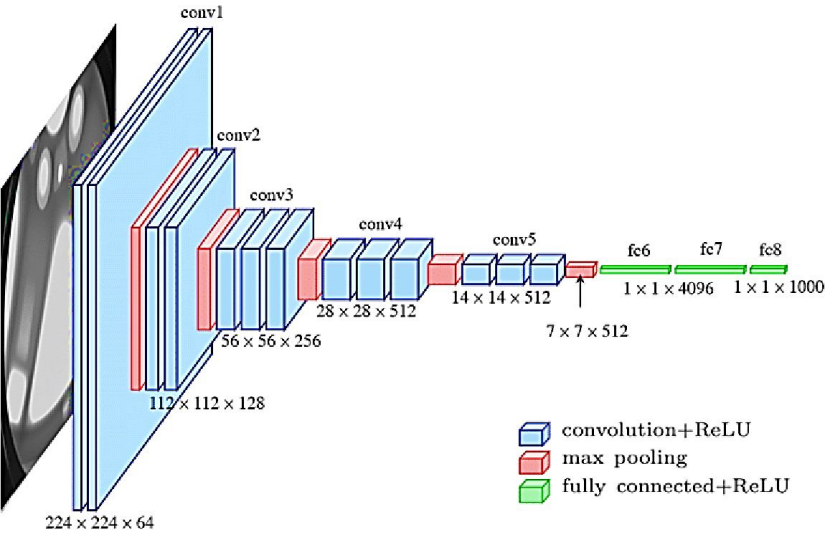
Actual: squamous_cell_carcinoma
Predicted: squamous_cell_carcinoma



Actual: benign
Predicted: benign



VGG-16



3 capas completamente conectadas

13 capas convolucionales + pooling

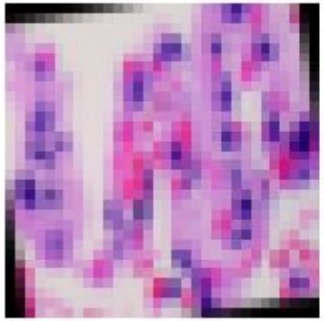
El tamaño de entrada para VGG16 es una imagen de 224x224 píxeles con 3 canales de color (RGB), lo que da lugar a una entrada de dimensión (224, 224, 3).

VGG-16

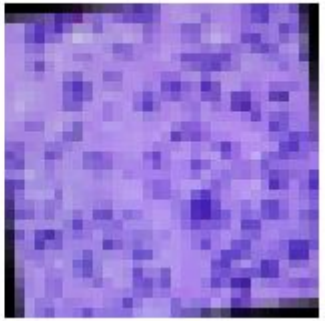
```
[ ]  # Testing the model
     model.eval()  # Set the model to evaluation mode
     correct = 0
     total = 0
     with torch.no_grad():
         for data in val_loader:
             images, labels = data[0].to(device), data[1].to(device)
             outputs = model(images)
             _, predicted = torch.max(outputs, 1)
             total += labels.size(0)
             correct += (predicted == labels).sum().item()

     print(f'Accuracy of the model on the 10,000 test images: {100 * correct / total:.2f}%')
```

```
Accuracy of the model on the 10,000 test images: 32.57%
```

Actual: squamous_cell_carcinoma
Predicted: squamous_cell_carcinoma



Actual: benign
Predicted: squamous_cell_carcinoma



GoogleNet

Training

```
# Ciclo de entrenamiento
num_epochs=20
for epoch in range(num_epochs):  # 20 épocas de entrenamiento
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        if i % 200 == 199:    # Imprimir cada 200 lotes
            print(f'[Epoch {epoch + 1}, Lote {i + 1}] Pérdida: {running_loss / 200:
            running_loss = 0.0

print('Entrenamiento finalizado')
```

```
[Epoch 1, Lote 200] Pérdida: 0.371
[Epoch 2, Lote 200] Pérdida: 0.258
[Epoch 3, Lote 200] Pérdida: 0.219
[Epoch 4, Lote 200] Pérdida: 0.182
[Epoch 5, Lote 200] Pérdida: 0.179
[Epoch 6, Lote 200] Pérdida: 0.141
[Epoch 7, Lote 200] Pérdida: 0.169
[Epoch 8, Lote 200] Pérdida: 0.153
[Epoch 9, Lote 200] Pérdida: 0.099
[Epoch 10, Lote 200] Pérdida: 0.124
[Epoch 11, Lote 200] Pérdida: 0.087
[Epoch 12, Lote 200] Pérdida: 0.086
[Epoch 13, Lote 200] Pérdida: 0.061
[Epoch 14, Lote 200] Pérdida: 0.064
[Epoch 15, Lote 200] Pérdida: 0.055
[Epoch 16, Lote 200] Pérdida: 0.052
[Epoch 17, Lote 200] Pérdida: 0.050
[Epoch 18, Lote 200] Pérdida: 0.046
[Epoch 19, Lote 200] Pérdida: 0.044
[Epoch 20, Lote 200] Pérdida: 0.024
Entrenamiento finalizado
```

Testing

```
# Testing the model
model.eval()  # Set the model to evaluation mode
correct = 0
total = 0
with torch.no_grad():
    for data in val_loader:
        images, labels = data[0].to(device), data[1].to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f'Accuracy of the model on the 10,000 test images: {100 * correct / total:.2f}%')
```

```
Accuracy of the model on the 10,000 test images: 97.53%
```

Actual: squamous_cell_carcinoma
Predicted: adenocarcinoma

Actual: squamous_cell_carcinoma
Predicted: squamous_cell_carcinoma

COMPARATIVA

	Modelo	Cant. Imágenes	Accuracy (%)
0	ResNet	10000	98.20
1	VGG-16	10000	32.57
2	Google Net	10000	97.53

Como se puede observar ResNet es superior a Google Net para este procesamiento; esto puede sostenerse al manejo de las conexiones residuales para la solución del desvanecimiento del gradiente, aunque para este caso no se observa una diferencia tan clara como con VGG-16; Resnet en este caso puede profundizar mucho más que VGG-16 sin perder la precisión, algo que es una limitante para VGG-16 cuando se trata de redes más profundas