TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TPHCM

KHOA CÔNG NGHỆ THÔNG TIN

_____

BÁO CÁO BÀI TẬP THỰC HÀNH II

PHÂN LỚP BẰNG MẠNG NƠ RON

Giảng viên:

- *Dương Nguyễn Thái Bảo*
- *Nguyễn Ngọc Đức*
- *Nguyễn Tiến Huy*
- *Bùi Tiến Lên*
- *Lê Thanh Phong*

Nhóm sinh viên thực hiện:

1. *Lê Thanh Viễn - 18120647*
2. *Đặng Văn Hiển - 18120363*
3. *Ngô Đăng Quang - 18120526*
4. *Nguyễn Đăng Quang - 18120527*

1.Thông tin nhóm:

1.1: Thông tin thành viên:

MSSV	Họ Tên	Email
18120647	Lê Thanh Viễn	18120647@student.hcmus.edu.vn
18120363	Đặng Văn Hiển	18120363@student.hcmus.edu.vn
18120526	Ngô Đăng Quang	18120526@student.hcmus.edu.vn
18120527	Nguyễn Đăng Quang	18120527@student.hcmus.edu.vn

1.2: Mức độ hoàn thành công việc:

Người thực hiện	Công việc	Mức độ hoàn thành
Lê Thanh Viễn	Phân tích bài toán, tìm tài liệu,..	100%
Đặng Văn Hiển	Cài đặt mạng nơ rơn, huấn luyện model,..	100%
Ngô Đăng Quang	Viết báo cáo	20%
Nguyễn Đăng Quang	Viết báo cáo	20%

2. Phân tích bài toán:

Trong nộng nghiệp, năng suất của cây trồng phụ thuộc rất nhiều vào thời tiết, chế độ chăm sóc, sâu bệnh,… để tăng năng suất người trồng cần có những biện pháp hạn chế những tác động của các yếu tố trên. Tuy nhiên, bản thân cây trồng cũng có những loại bệnh nhất định và thường biểu hiện ra bên ngoài trên thân, lá… những biểu hiện đó bất thường trên cây trồng thường liện quan tới một loại bệnh nào đó.

Nhằm giúp người trồng xác định được loại bệnh đang có trên cây trồng, sinh viên được cung cấp tập dữ liệu hình ảnh về các loại bệnh trên cây và tập dữ liệu đã được xác định được loại bệnh (train.csv và test.csv) bao gồm 4 loại:

- Combinations
- Healthy
- Rust
- Scab

- Dữ liệu chứa trong các tập tin:
    - sample_submission.csv
        - image_id – khoá chính, cũng tên file ảnh
        - healthy, multiple_diseases, rust, scab – ghi dự đoán xác suất
    - test.csv
        - image_id – khoá chính, cũng tên file ảnh
    - train.csv
        - image_id – khoá chính, cũng tên file ảnh
        - healthy, multiple_diseases, rust, scab –

3. Chuẩn bị dữ liệu và tiền xử lý:

Tải về và giải nén dữ liệu từ link google drive thầy cung cấp:

Download data

```
[4]  # Download Datasets plant-pathology-2020-fgvc7.zip by id "1Eg_0c3mvQCu0SpVjLdik1-g1Tuz22ObS"
     isDone = os.path.isfile( "/content/sample_submission.csv")
     if not isDone:
       !gdown --id "1Eg_0c3mvQCu0SpVjLdik1-g1Tuz22ObS"
       !unzip "/content/plant-pathology-2020-fgvc7.zip"
       !rm "/content/plant-pathology-2020-fgvc7.zip"
```

Thiết đặt đường dẫn và các thông số chính

Settings

```
[5]  # Define path
     IMAGES_PATH = '/content/images/'
     TRAIN_PATH = '/content/train.csv'
     TEST_PATH = '/content/test.csv'
     SAMPLE_SUBMISSION_PATH = '/content/sample_submission.csv'
```

```
[6]  # Read csv
     df_train = pd.read_csv(TRAIN_PATH)
     df_test = pd.read_csv(TEST_PATH)
     df_submission = pd.read_csv(SAMPLE_SUBMISSION_PATH)
     df_submission.iloc[:, 1:] = 0
```

```
[7]  # Configuration for training workflow
     N_EPOCHS = 5
     BATCH_SIZE = 8

     INPUT_SIZE = 224
     IMG_SHAPE = (1365, 2048, 3)

     _num_classes = 4
     _num_workers = 0
     _alpha = 1e-3
     _betas=(0.9, 0.999)
     _eps=1e-08
```

```
[8]  # Set TPU device
     device = xm.xla_device()
     torch.set_default_tensor_type('torch.FloatTensor')
```

Cài đặt lớp lấy dữ liệu (ảnh) :

Class Dataset

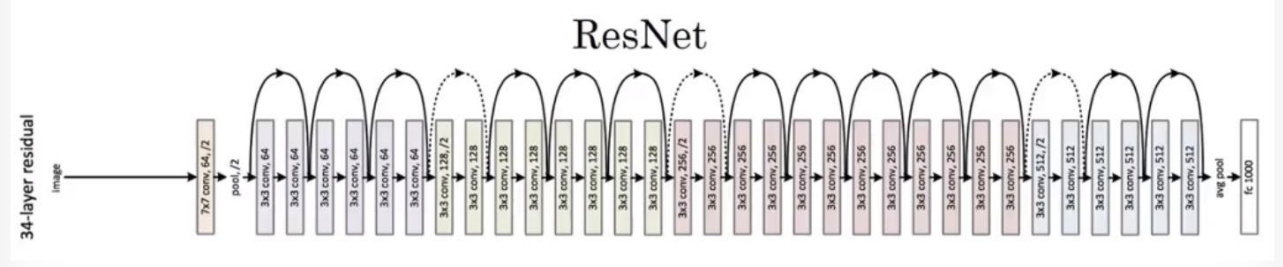
```
[9]  # Dataset
     class IMGDataset(Dataset):

         def __init__(self, df, transforms=None):
             self.df = df
             self.transforms=transforms

         def __len__(self):
             return self.df.shape[0]

         def __getitem__(self, idx):
             # Get image
             image = cv2.imread(IMAGES_PATH + self.df.loc[idx, 'image_id'] + '.jpg', cv2.IMREAD_COLOR)
             image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

             # Get labels
             labels = self.df.loc[idx, ['healthy', 'multiple_diseases', 'rust', 'scab']].values
             labels = torch.from_numpy(labels.astype(np.int8))
             labels = labels.unsqueeze(-1)

             # transform ('random')
             if self.transforms:
                 image = self.transforms(image)
                 image = image.to(device, dtype=torch.float)

             return image, labels
```

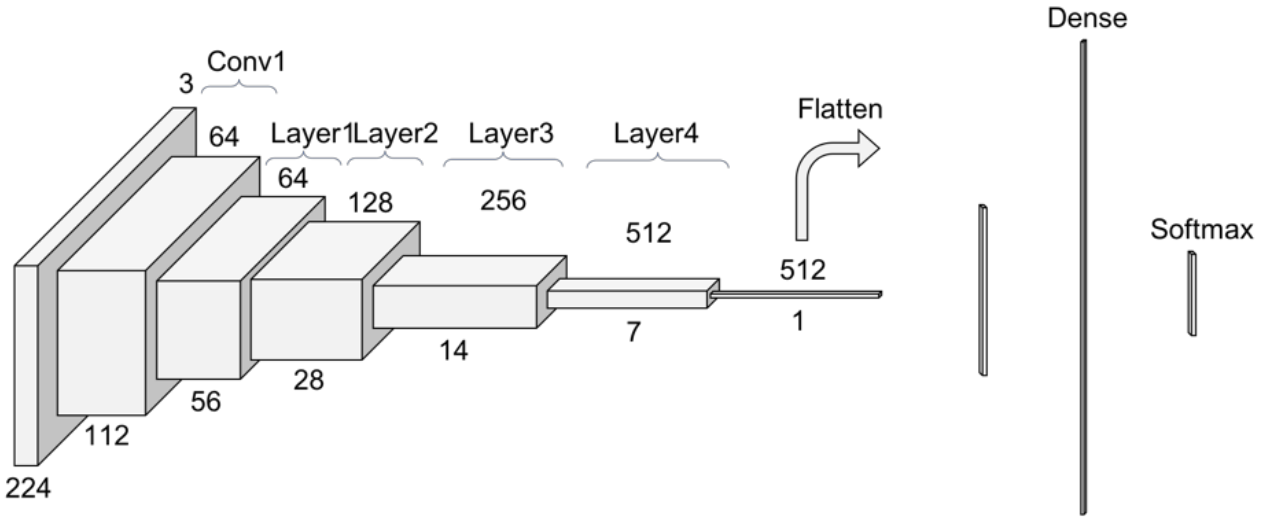
- Cài đặt các phương pháp biến đổi ảnh ngẫu nhiên phục vụ việc học của mạng nơ ron : random crop, random flip, random rotation, …

```
# Train transformation ("random")
trans_train = T.Compose([
    T.ToPILImage(),
    T.RandomResizedCrop(size=INPUT_SIZE),
    T.RandomHorizontalFlip(p=0.5),
    T.RandomVerticalFlip(p=0.5),
    T.RandomRotation(degrees=30),
    T.ToTensor(),
    T.ColorJitter(brightness=0.2, contrast=0.2),
    T.GaussianBlur(kernel_size=3, sigma=(0.1, 2.0)),
    T.Normalize(mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225] )
    ])

# Validation transformation
trans_valid = T.Compose([
    T.ToPILImage(),
    T.Resize(size=INPUT_SIZE),
    T.ToTensor()
    ])
```

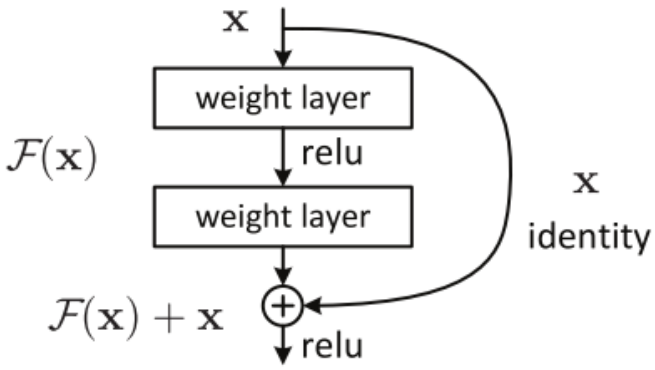
4. Cài đặt thuật toán mạng nơ-ron (ResNet34):

ResNet (Residual Network) được giới thiệu đến công chúng vào năm 2015 và thậm chí đã giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Không những thế nó còn đứng vị trí đầu tiên trong cuộc thi ILSVRC and COCO 2015 với ImageNet Detection, ImageNet localization, Coco detection và Coco segmentation.





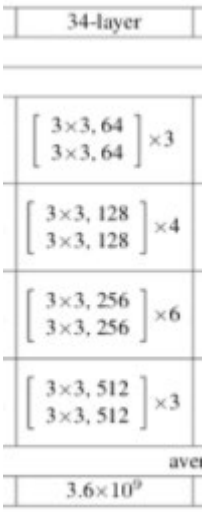
Kiến trúc block có các "đường tắt" qua một hay nhiều lớp tạo thành các khối Residual Block (Basic block) còn lại thì nó gần như các mạng khác gồm có convolution, pooling, activation và fully-connected layer.



- Viết code ResNet34

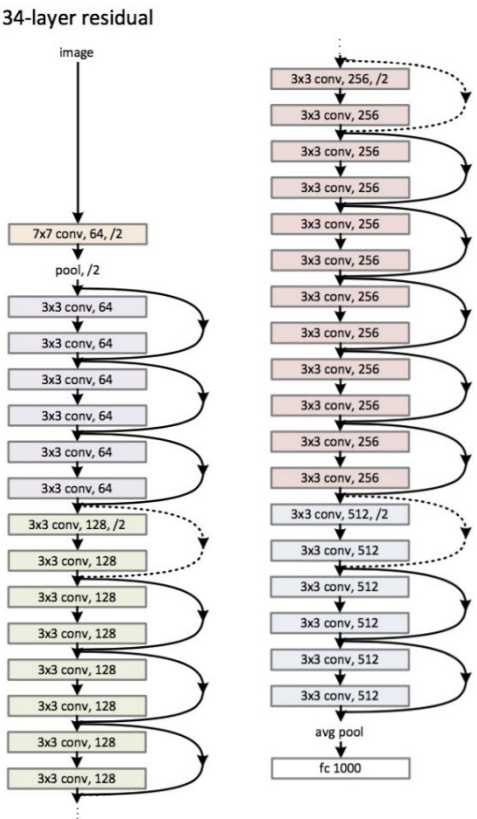
```
class ResNet34(nn.Module):
    def __init__(self,num_classes):
        super(ResNet34,self).__init__()

        self.conv1=nn.Conv2d(3,64,kernel_size=7,stride=2,padding=3,bias=False)
        self.bn1=nn.BatchNorm2d(64)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool=nn.MaxPool2d(kernel_size=3,stride=2,padding=1)

        self.n_inplanes=64
        self.layer1=self.make_layers(64,3,stride=1)
        self.layer2=self.make_layers(128,4,stride=2)
        self.layer3=self.make_layers(256,6,stride=2)
        self.layer4=self.make_layers(512,3,stride=2)

        self.avgpool=nn.AdaptiveAvgPool2d((1,1))
        self.fc=nn.Linear(512,num_classes)

    # Make layer
    def make_layers(self,n_planes, blocks,stride):
        # first planes downsample ?
        downsample = None
        if stride != 1 or self.n_inplanes != n_planes:
            downsample = nn.Sequential(
                nn.Conv2d(self.n_inplanes, n_planes , stride),
                nn.BatchNorm2d(n_planes)
                )

        layers = []
        layers.append(BasicBlock(self.n_inplanes, n_planes, stride, downsample))
        self.n_inplanes = n_planes
        for _ in range(1, blocks):
            layers.append(BasicBlock(n_planes, n_planes,stride))

        return nn.Sequential(*layers)
```

```
def forward(self,x):
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)

    return x
```



- Viết code Residual Block (Basic block)

```
class BasicBlock(nn.Module):
    def __init__( self, n_inplanes, n_planes, stride, downsample=None ):
        super(BasicBlock, self).__init__()
        self.downsample = downsample

        self.conv1 = nn.Conv2d(n_inplanes, n_planes,3, stride)
        self.bn1 = nn.BatchNorm2d(n_planes)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(n_planes, n_planes,3,1)
        self.bn2 = nn.BatchNorm2d(n_planes)


    def forward(self, x):
        identity = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        # shortcut : D
        if self.downsample is not None:
            identity = self.downsample(x)
        print(out)
        print(identity)
```



5. Huấn luyện mô hình:

- Nạp dữ liệu, transformers vào Dataset => Dataloader
- Loss funciton (với classification) là Cross Entropy One-hot
- Optimizer (gradient descent optimization algorithms) là Adam

```
# Init model
model = ResNet34(num_classes=_num_classes)
model.to(device)

# Split train set and valid set
sub_train, sub_valid = train_test_split(df_train, test_size=0.2, random_state=42)
sub_train.reset_index(drop=True, inplace=True)
sub_valid.reset_index(drop=True, inplace=True)

# Set Dataset
dataset_train = IMGDataset(df=sub_train, transforms=trans_train)
dataset_valid = IMGDataset(df=sub_valid, transforms=trans_valid)
# Set Dataloader
dataloader_train = DataLoader(dataset_train, batch_size=BATCH_SIZE, num_workers=0, shuffle=True, pin_memory=True, drop_last=True)
dataloader_valid = DataLoader(dataset_valid, batch_size=BATCH_SIZE, num_workers=0, shuffle=False, pin_memory=True, drop_last=False)

# Set Optimizer - gradient descent optimization algorithms
lossFunc = nn.CrossEntropyLoss()

# Set Optimizer - gradient descent optimization algorithms
optimizer = optim.Adam(model.parameters(),lr=_alpha, betas=_betas , eps=_eps)

# Train
train_results = Train( model, lossFunc, optimizer, dataloader_train, dataloader_valid)
```

- Lần lượt train, .. zero_grad(), forward(), backward(), optimer_strep(), … eval() để huấn luyện mô hình mỗi epoch..

```
def Train( model, lossFunc, optimizer, dataloader_train, dataloader_valid):

    for epoch in tqdm(range(N_EPOCHS)):
        # print information
        print('  ' + ('-' * 20))
        print('  Epoch {}/{}'.format(epoch + 1, N_EPOCHS))

        # call model
        model.train()

        # looping batch
        train_loss = 0
        for step, (images,labels)  in tqdm(enumerate(dataloader_train)):
            images = images.to(device)
            labels = labels.to(device)

            # zero gradient
            optimizer.zero_grad()

            # forward
            outputs = model(images)

            # get loss
            value = one_hot_to_values(labels.squeeze(-1))
            loss = lossFunc(outputs, value)
            # backward pass
            loss.backward()
            train_loss += loss.item()

            # updates
            xm.optimizer_step(optimizer, barrier=True)

        # Validate
        model.eval()

        # init validation loss, predicted and labels
        valid_loss = 0
```

6. Đánh giá mô hình mạng vừa huấn luyện:

Chạy huấn luyện từ các batch nhỏ mỗi epoch và in ra train_loss, valid_loss, accuracy.

```
[18]  ----------
      Epoch 1/5
      182/? [12:12<00:00, 4.03s/it]
      46/? [10:08<00:00, 13.23s/it]
Epoch 1: train_loss=0.96400839, valid_loss=1.34471441, acc=0.61558016
      ----------
      Epoch 2/5
      182/? [09:04<00:00, 2.99s/it]
      46/? [02:07<00:00, 2.76s/it]
Epoch 2: train_loss=0.75424034, valid_loss=1.81153155, acc=0.63382748
      ----------
      Epoch 3/5
      182/? [06:15<00:00, 2.07s/it]
      46/? [04:36<00:00, 6.01s/it]
Epoch 3: train_loss=0.72855739, valid_loss=1.17352783, acc=0.74445737
      ----------
      Epoch 4/5
      182/? [03:41<00:00, 1.22s/it]
      46/? [02:03<00:00, 2.69s/it]
Epoch 4: train_loss=0.63121917, valid_loss=1.24013975, acc=0.67624195
      ----------
      Epoch 5/5
      182/? [02:04<00:00, 1.46it/s]
      46/? [00:26<00:00, 1.75it/s]
Epoch 5: train_loss=0.63411876, valid_loss=1.32889840, acc=0.61422406
```

```
[18] train_results = pd.DataFrame(train_results)
     train_results.head()
```

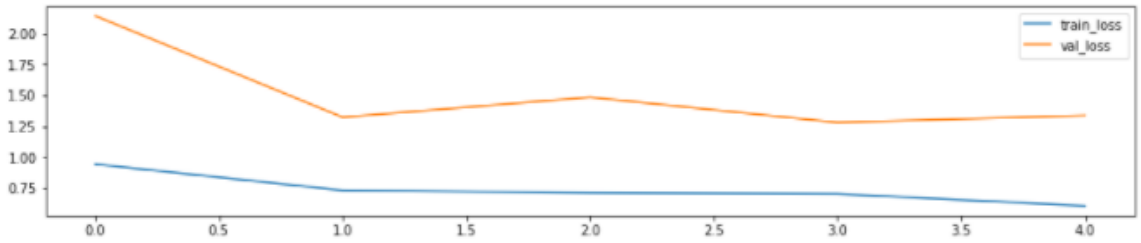
	epoch	train_loss	valid_loss	accuracy
0	0	0.944772	2.140403	0.607331
1	1	0.733421	1.323166	0.686648
2	2	0.713136	1.484584	0.647843
3	3	0.705219	1.280847	0.694805
4	4	0.608746	1.337493	0.665281

```
[19] train_results.to_csv('train_result.csv')
     files.download("train_result.csv")
```

▾ Training loss

```
[20] def display_training_loss(train_result):
        plt.figure(figsize=(15,10))
        plt.subplot(3,1,1)
        train_loss = train_result['train_loss']
        plt.plot(train_loss.index, train_loss, label = 'train_loss')
        plt.legend()
        val_loss = train_result['valid_loss']
        plt.plot(val_loss.index, val_loss, label = 'val_loss')
        plt.legend()
```

```
[21] display_training_loss(train_results)
```

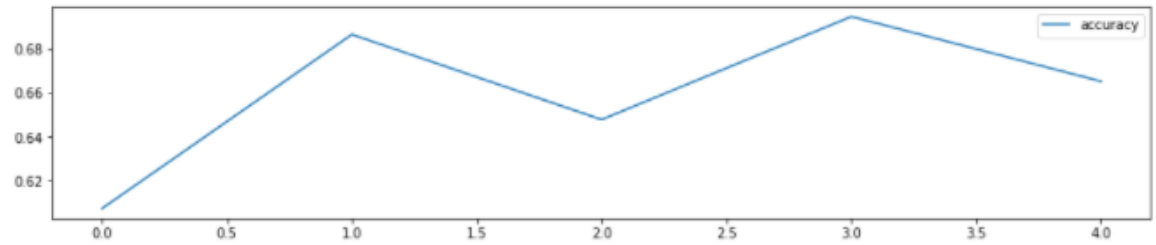


▾ Validation score

```
def display_accuracy(train_result):
    plt.figure(figsize=(15,10))
    plt.subplot(3,1,1)
    accuracy = train_result['accuracy']
    plt.plot(accuracy.index, accuracy, label = 'accuracy')
    plt.legend()
```

+ Mã + Văn bản

```
[23] display_accuracy(train_results)
```



7. Tham khảo:

- https://pytorch.org/vision/0.8/models.html
- https://towardsdatascience.com/top-10-cnn-architectures-every-machine-learning-engineer-should-know-68e2b0e07201