1. **Phân nhóm bệnh cà chua theo nguyên nhân gây bệnh**

Các bệnh trên cây cà chua có thể được phân thành **4 nhóm chính**:

**1. Nhóm bệnh do nấm (Fungal diseases)**

* Early Blight (Alternaria)
* Late Blight (Phytophthora)
* Septoria Leaf Spot
* Leaf Mold
* Powdery Mildew

**2. Nhóm bệnh do vi khuẩn (Bacterial diseases)**

* Bacterial Spot
* Bacterial Speck
* Bacterial Canker

**3. Nhóm bệnh do virus (Viral diseases)**

* Tomato Mosaic Virus (TMV)
* Yellow Leaf Curl Virus
* Tomato Spotted Wilt Virus

**4. Nhóm rối loạn sinh lý (Physiological disorders)**

* Blossom End Rot
* Sunscald
* Nutrient deficiencies

**Hệ thống MoE**

**Cách tiếp cận tối ưu: 3 experts chuyên biệt**

python

Copy Code

# Phân chia experts theo đặc điểm bệnh học:

Expert 1: Fungal Disease Specialist

- Chuyên xử lý: Early Blight, Late Blight, Septoria Leaf Spot, Leaf Mold

- Đặc trưng: Các vết đốm, mốc, thay đổi màu sắc lá

Expert 2: Bacterial & Viral Disease Specialist

- Chuyên xử lý: Bacterial Spot, Bacterial Speck, Mosaic Virus

- Đặc trưng: Các vết loang, biến dạng lá, thay đổi cấu trúc

Expert 3: Physiological & Healthy Classifier

- Chuyên xử lý: Blossom End Rot, Healthy leaves, các rối loạn dinh dưỡng

- Đặc trưng: Lá khỏe mạnh và các bất thường không do bệnh

**Cấu hình MoE được đề xuất:**

python

Copy Code

# Gating strategy

def disease\_router(image\_features):

# Phân tích đặc trưng chính

spot\_density = analyze\_spot\_patterns(image\_features)

color\_variance = analyze\_color\_distribution(image\_features)

shape\_deformation = analyze\_leaf\_shape(image\_features)

if spot\_density > threshold\_1:

return "fungal\_expert" # Expert 1

elif shape\_deformation > threshold\_2:

return "bacterial\_viral\_expert" # Expert 2

else:

return "physiological\_expert" # Expert 3

**Đánh giá hiệu quả:**

python

Copy Code

# Metrics để đánh giá

metrics = {

'overall\_accuracy': accuracy\_score(y\_true, y\_pred),

'per\_group\_f1': {

'fungal': f1\_score(fungal\_true, fungal\_pred),

'bacterial\_viral': f1\_score(bv\_true, bv\_pred),

'physiological': f1\_score(phys\_true, phys\_pred)

},

'expert\_utilization': calculate\_expert\_usage(),

'confusion\_reduction': compare\_confusion\_matrices()

}



**Các mô hình ML cụ thể cho từng expert**

**Expert 1: Fungal Disease Specialist**

**Khuyến nghị: ResNet50/EfficientNet + CNN chuyên biệt**

python

Copy Code

# Architecture cho Fungal Expert

class FungalExpert(nn.Module):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        # Backbone: EfficientNet-B3 (tối ưu cho texture analysis)

        self.backbone = EfficientNet.from\_pretrained('efficientnet-b3')

        # Specialized layers cho spot detection

        self.spot\_detector = nn.Sequential(

            nn.Conv2d(1536, 512, 3, padding=1),

            nn.BatchNorm2d(512),

            nn.ReLU(),

            nn.Conv2d(512, 256, 3, padding=1),

            nn.AdaptiveAvgPool2d(1)

        )

        # Classifier cho 4 loại bệnh nấm + healthy

        self.classifier = nn.Linear(256, 5)

**Lý do lựa chọn:**

* **EfficientNet**: Xuất sắc trong việc phát hiện texture patterns và spot characteristics
* **Specialized convolution layers**: Tối ưu cho việc phát hiện các vết đốm tròn, có viền

**Expert 2: Bacterial & Viral Disease Specialist**

**Khuyến nghị: Vision Transformer (ViT) + Attention mechanism**

python

Copy Code

# Architecture cho Bacterial/Viral Expert

class BacterialViralExpert(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Vision Transformer cho global pattern recognition

self.vit = VisionTransformer(

image\_size=224,

patch\_size=16,

num\_classes=4, # 3 diseases + healthy

dim=768,

depth=12,

heads=12,

mlp\_dim=3072

)

# Multi-scale attention cho deformation detection

self.deformation\_attention = MultiScaleAttention(

dim=768,

scales=[1, 2, 4]

)

**Lý do lựa chọn:**

* **Vision Transformer**: Xuất sắc trong việc capture global patterns và spatial relationships
* **Attention mechanism**: Phát hiện được subtle deformations và irregular patterns

**Expert 3: Physiological & Healthy Classifier**

**Khuyến nghị: MobileNetV2 + Traditional ML ensemble**

python

Copy Code

# Architecture cho Physiological Expert

class PhysiologicalExpert(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Lightweight CNN cho feature extraction

self.feature\_extractor = MobileNetV2(pretrained=True)

self.feature\_extractor.classifier = nn.Identity()

# Traditional ML ensemble cho physiological analysis

self.ml\_ensemble = {

'svm': SVC(kernel='rbf', probability=True),

'random\_forest': RandomForestClassifier(n\_estimators=100),

'gradient\_boost': GradientBoostingClassifier()

}

# Meta-learner

self.meta\_classifier = LogisticRegression()

**Lý do lựa chọn:**

* **MobileNetV2**: Lightweight, hiệu quả cho việc phân biệt healthy vs unhealthy
* **Traditional ML ensemble**: Tốt cho physiological disorders vì có ít training data
* **Meta-learning**: Kết hợp ưu điểm của deep learning và traditional ML

**Gating Network Architecture**

**Khuyến nghị: Lightweight CNN + Decision Tree**

python

Copy Code

class SmartGatingNetwork(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Lightweight feature extractor

self.feature\_net = nn.Sequential(

nn.Conv2d(3, 32, 7, stride=2),

nn.BatchNorm2d(32),

nn.ReLU(),

nn.AdaptiveAvgPool2d(4),

nn.Flatten(),

nn.Linear(32\*16, 128),

nn.ReLU(),

nn.Linear(128, 64)

)

# Rule-based routing với learned weights

self.router = nn.Sequential(

nn.Linear(64, 32),

nn.ReLU(),

nn.Linear(32, 3), # 3 experts

nn.Softmax(dim=1)

)

**Training Strategy cho từng Expert**

**1. Fungal Expert Training:**

python

Copy Code

# Specialized data augmentation cho fungal diseases

fungal\_transforms = transforms.Compose([

transforms.RandomRotation(30),

transforms.ColorJitter(brightness=0.3, contrast=0.3),

transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),

# Custom augmentation cho spot enhancement

SpotEnhancementAugmentation(),

transforms.ToTensor(),

transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

])

# Loss function với class weighting

fungal\_loss = nn.CrossEntropyLoss(weight=torch.tensor([1.0, 2.0, 1.5, 1.8, 0.8]))

**2. Bacterial/Viral Expert Training:**

python

Copy Code

# Augmentation tập trung vào shape deformation

bv\_transforms = transforms.Compose([

transforms.RandomAffine(degrees=15, translate=(0.1, 0.1), shear=10),

transforms.RandomPerspective(distortion\_scale=0.2),

ElasticTransform(), # Custom elastic deformation

transforms.ToTensor(),

transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

])

# Focal loss cho imbalanced classes

bv\_loss = FocalLoss(alpha=0.25, gamma=2.0)

**3. Physiological Expert Training:**

python

Copy Code

# Minimal augmentation để preserve natural characteristics

phys\_transforms = transforms.Compose([

transforms.Resize(224),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

])

# Ensemble training

def train\_physiological\_ensemble(X\_train, y\_train):

# Train individual models

svm\_model.fit(X\_train, y\_train)

rf\_model.fit(X\_train, y\_train)

gb\_model.fit(X\_train, y\_train)

# Meta-learning

meta\_features = np.column\_stack([

svm\_model.predict\_proba(X\_train),

rf\_model.predict\_proba(X\_train),

gb\_model.predict\_proba(X\_train)

])

meta\_classifier.fit(meta\_features, y\_train)

**Tips**

**1. Transfer Learning Strategy:**

python

Copy Code

# Pre-trained weights cho từng expert

fungal\_expert.load\_state\_dict(torch.load('efficientnet\_plant\_pretrained.pth'))

bv\_expert.load\_state\_dict(torch.load('vit\_deformation\_pretrained.pth'))

phys\_expert.load\_state\_dict(torch.load('mobilenet\_health\_pretrained.pth'))

**2. Expert Specialization Loss:**

python

Copy Code

def specialization\_loss(expert\_outputs, true\_labels, expert\_id):

"""Encourage experts to specialize on their assigned disease groups"""

base\_loss = nn.CrossEntropyLoss()(expert\_outputs, true\_labels)

# Penalty for confident predictions on wrong disease groups

wrong\_group\_penalty = calculate\_wrong\_group\_confidence(expert\_outputs, expert\_id)

return base\_loss + 0.1 \* wrong\_group\_penalty



Cách tiếp cận này sẽ tạo ra sự đa dạng thực sự giữa các experts và cải thiện đáng kể hiệu suất tổng thể của hệ thống MoE.[object Object]