# GoogLeNet (Inception)

## <Going Deeper with Convolutions>

서울과학기술대학교 국방인공지능응용학과
이찬호

# Contents

## Part 1

이전 QnA

## Part 2

GoogLeNet 구현

# Part 1

## 이전 QnA

Part 1

# 이전 QnA

## Auxiliary Classifier & Architecture

- Auxiliary Classifier loss에 0.3을 곱한 이유
  - **구체적인 명시 없음**
  - 여러 훈련을 통해 0.3 값을 **도출**한 것으로 보임
- 실제 테스트 시 Auxiliary Classifier를 제거한 이유
  - **구체적인 명시 없음**
  - 추론 시 **불필요한 계산을 줄여 효율성을 높이고자 삭제**한 것으로 보임
- Architecture – +1(S), +3(V)의 뜻
  - **구체적인 명시 없음**
  - +1(S) = Stride가 1
  - +3(V) = Valid Padding이 3 pixel씩 이동

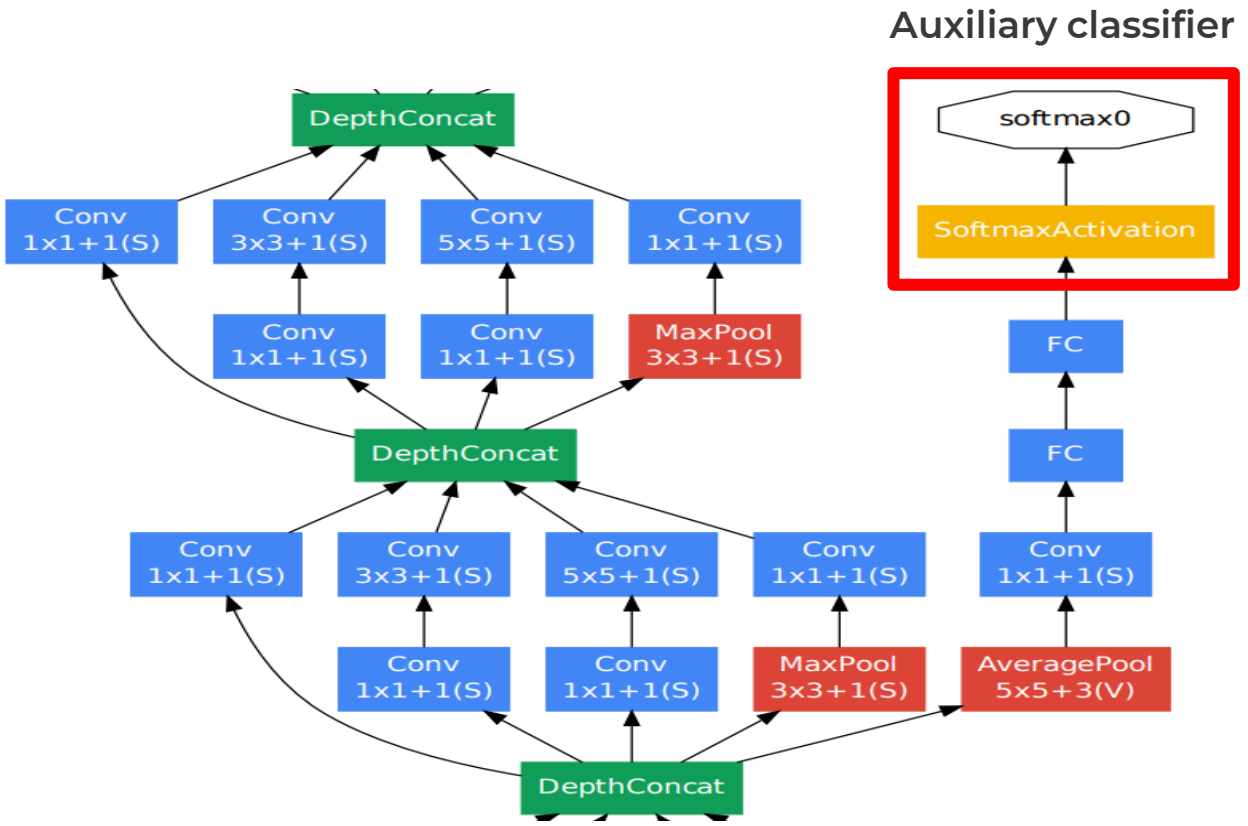



Fig 1. GoogLeNet Architecture 일부

Part 1

# 이전 QnA

| 모델명 | VGG (2014) | (GoogLeNet) Inception V1 (2014) | ResNet (2015) | Inception V4 (2016) |
|---|---|---|---|---|
| 특징 | • **3X3** 필터로 간단하며 깊은 아키텍처 구성<br>• 작은 필터를 여러 번 적용하여 큰 필터를 한 번 사용하는 것보다 효과적임을 보여줌<br>• 학습 속도 향상을 위해 작은 필터와 작은 **Stride** 사용 | • **Inception module**을 도입하여 네트워크 깊이를 늘리고 계산량을 줄임<br>• **1X1, 3X3, 5X5** 필터와 **3X3 Max Pooling**을 병렬로 사용하는 **Inception module**<br>• **Auxiliary Classifier**를 사용하여 기울기 소실 완화 | • **Residual Connection**을 도입하여 기울기 소실 문제 해결<br>• **Input** 값을 **Output** 값에 더하는 방식으로 정보 손실 방지 | • 기존 모델들 **(~V3)**의 구조가 복잡하여 필터 분포를 조절하고 연결방식을 최적화함<br>• **ResNet**의 **Residual Connection** 방식을 도입하여 성능 향상<br>• **4**개의 **Inception-ResNet module**과 **3**개의 **Inception module**로 구성 |
| 장점 | • 구조가 단순하여 구현하기 쉬움<br>• **3X3** 필터를 사용하여 **Receptive Field**를 확보 | • **Inception module**로 네트워크 깊이를 늘리고 계산량을 줄임<br>• **Auxiliary Classifier**로 기울기 소실 문제 완화 | • **Residual Connection**으로 기울기 소실 문제 해결하여 깊은 네트워크 학습 가능<br>• 간단하고 일반적인 구조 | • **Inception module**과 **Residual Connection**의 결합으로 성능 향상 |
| 단점 | • 매개변수 많음<br>• 메모리 사용량 증가 | • **Inception** 구조가 복잡하여 설계 및 최적화 어려움<br>• 전체적인 네트워크 구조가 복잡함 | • 네트워크가 깊어질수록 계산 복잡도와 메모리 사용량 증가 | • 구조가 매우 복잡<br>• 매개변수 많음<br>• 메모리 사용량 증가 |

# Part 2

## GoogLeNet 구현

Part 2

# GoogLeNet 구현

## DATASET

- ImageNet 2014 Classification Dataset은 2012와 변경점이 없음
- Class명이 n(숫자)로 구성 (일반적인 명칭 X)
- Dataset은 아래와 같이 구성
- Train Dataset
  - 1,000 Classes
  - 1,281,167 Images
- Valid Dataset
  - 1,000 Classes
  - 50,000 Images

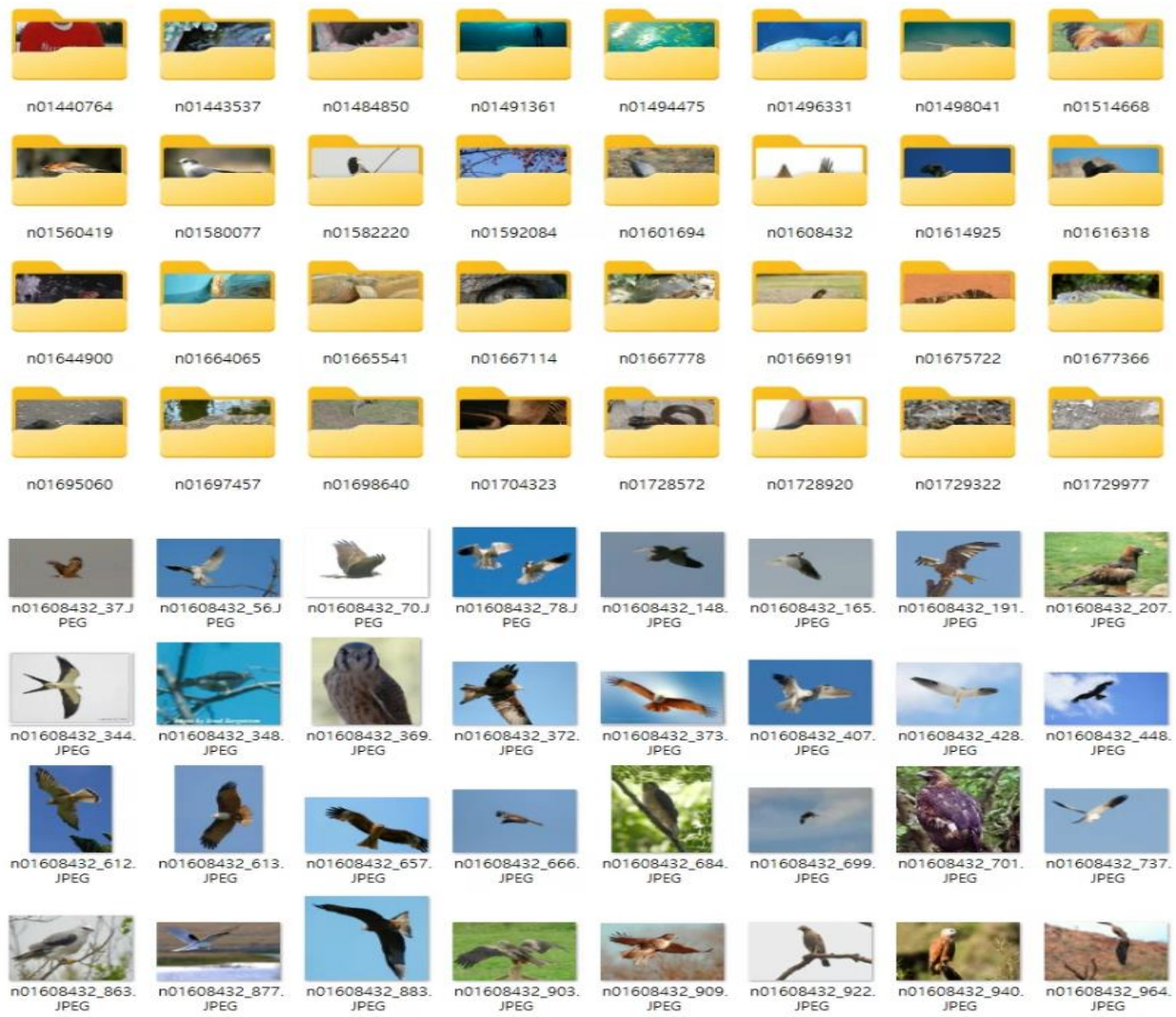



**Fig 2. Train Class 일부 (위) & Img 중 일부 (아래)**

Part 2

# GoogLeNet 구현

## Inception Module & Auxiliary Classifier 코드

- Inception Module 코드만 소개
- Input으로 x, filters를 받음
- Path1 – 1X1 Conv layer
  - Input의 공간적인 특징 유지
- Path2 – 1X1 Conv layer 다음 3X3 Conv layer
  - 더 넓은 영역의 특징 감지
- Path3 – 1X1 Conv layer 다음 5X5 Conv layer
  - 더 넓은 영역의 특징 감지
- Path4 – 3X3 Max Pooling layer 다음 1X1 Conv layer
  - 공간의 차원을 줄이고 Pooling을 통해 특징을 집중시킴(특징을 모은다)
- 4개 Path의 다양한 크기의 필터를 Concatenate layer를 사용하여 결합

```python
def inception(x, filters):
    # 1x1
    path1 = Conv2D(filters=filters[0], kernel_size=(1,1), strides=1, padding='same', activation='relu')(x)

    # 1x1->3x3
    path2 = Conv2D(filters=filters[1][0], kernel_size=(1,1), strides=1, padding='same', activation='relu')(x)
    path2 = Conv2D(filters=filters[1][1], kernel_size=(3,3), strides=1, padding='same', activation='relu')(path2)

    # 1x1->5x5
    path3 = Conv2D(filters=filters[2][0], kernel_size=(1,1), strides=1, padding='same', activation='relu')(x)
    path3 = Conv2D(filters=filters[2][1], kernel_size=(5,5), strides=1, padding='same', activation='relu')(path3)

    # 3x3->1x1
    path4 = MaxPool2D(pool_size=(3,3), strides=1, padding='same')(x)
    path4 = Conv2D(filters=filters[3], kernel_size=(1,1), strides=1, padding='same', activation='relu')(path4)

    return Concatenate(axis=-1)([path1,path2,path3,path4])
```

```python
def auxiliary(x,name=None):
    layer=AveragePooling2D((5,5),strides=3,padding='valid')(x)
    layer=Conv2D(128,(1,1),1,padding='same',activation='relu')(layer)
    layer=Flatten()(layer)
    layer=Dense(256,activation='relu')(layer)
    layer=Dropout(0.4)(layer)
    layer=Dense(1000,activation='softmax',name=name)(layer)
    return layer
```

Fig 3. Inception Module & Auxiliary Classifier 코드

Part 2

# GoogLeNet 구현

## GoogLeNet 코드

- 7X7 Conv layer, 3X3 Max pooling, Batchnomarlization 적용하여 공간 차원 축소
- 1X1, 3X3 Conv layer를 적용하여 특징을 추출, 이후 Batchnomalizaion, Pooling layer을 통해 차원 축소
- Inception Module 사용, Max pooling을 이용해 차원 축소하며 Auxiliary Classifier를 추가해 기울기 소실 문제 완화
- Feature map을 1차원 벡터로 평탄화(Flatten)하며 Dropout, FC layer를 적용해 네트워크의 Feature 추출
- 1,000개 Class를 Softmax를 사용해 최종 예측을 수행

```
def GoogLeNet():
    layer_in = Input(shape=(224,224,3))

    layer = Conv2D(filters=64, kernel_size=(7,7), strides=2, padding='same', activation='relu')(layer_in)
    layer = MaxPooling2D(pool_size=(3,3), strides=2, padding='same')(layer)
    layer = BatchNormalization()(layer)

    layer = Conv2D(filters=64, kernel_size=(1,1), strides=1, padding='same', activation='relu')(layer)
    layer = Conv2D(filters=192, kernel_size=(3,3), strides=1, padding='same', activation='relu')(layer)
    layer = BatchNormalization()(layer)
    layer = MaxPooling2D(pool_size=(3,3), strides=2, padding='same')(layer)

    layer = inception(layer, [ 64,  (96,128), (16,32), 32])
    layer = inception(layer, [128, (128,192), (32,96), 64])
    layer = MaxPooling2D(pool_size=(3,3), strides=2, padding='same')(layer)

    layer = inception(layer, [192,  (96,208),  (16,48),  64])
    aux1  = auxiliary(layer, name='aux1')
    layer = inception(layer, [160, (112,224),  (24,64),  64])
    layer = inception(layer, [128, (128,256),  (24,64),  64])
    layer = inception(layer, [112, (144,288),  (32,64),  64])
    aux2  = auxiliary(layer, name='aux2')
    layer = inception(layer, [256, (160,320), (32,128), 128])
    layer = MaxPooling2D(pool_size=(3,3), strides=2, padding='same')(layer)

    layer = inception(layer, [256, (160,320), (32,128), 128])
    layer = inception(layer, [384, (192,384), (48,128), 128])
    layer = AveragePooling2D(pool_size=(7,7), strides=1, padding='valid')(layer)

    layer = Flatten()(layer)
    layer = Dropout(0.4)(layer)
    layer = Dense(units=256, activation='linear')(layer)
    main = Dense(units=1000, activation='softmax', name='main')(layer)

    model = Model(inputs=layer_in, outputs=[main, aux1, aux2])
    model.compile(optimizer=SGD(learning_rate=0.01), loss='categorical_crossentropy', metrics=['accuracy'])
    model.summary()

    return model
```

Fig 4. GoogLeNet 코드

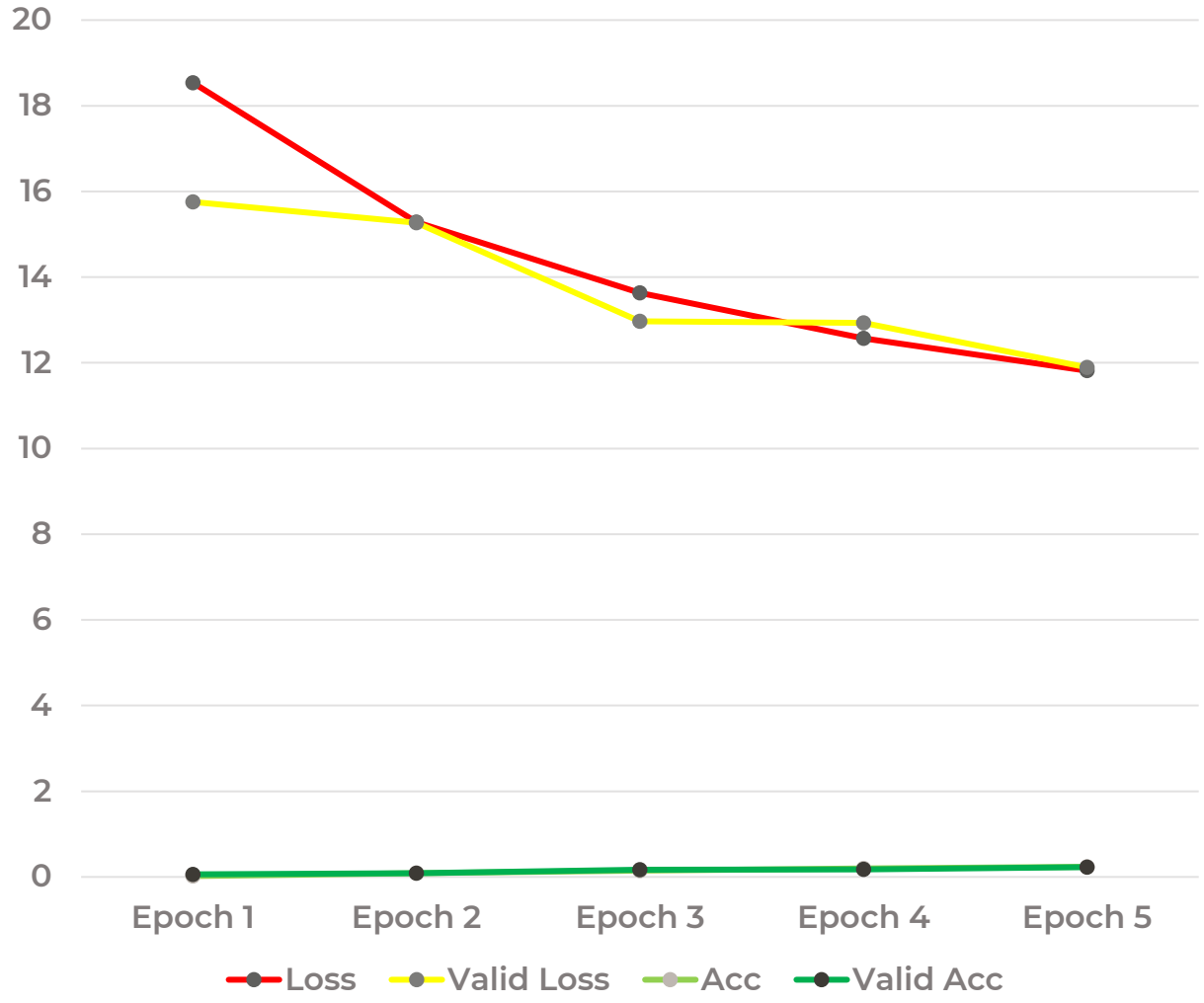
Part 2

# GoogLeNet 구현

## Train Method & Train Result

- Train Method
  - Optimizer – SGD
  - Learning Rate – 0.01
  - Loss – Categorical Crossentropy
  - Epoch – 10
- Train Result (5/10)
  - Loss – 11.816
  - Valid Loss – 11.89476
  - Accuracy – 0.2368
  - Valid Accuracy – 0.2286

Table 1. Train Result



# THANK YOU FOR YOUR ATTENTION