VGG net

with Adam & RMSprop

무인이동체공학과 **17011882** 김 우 혁

목차

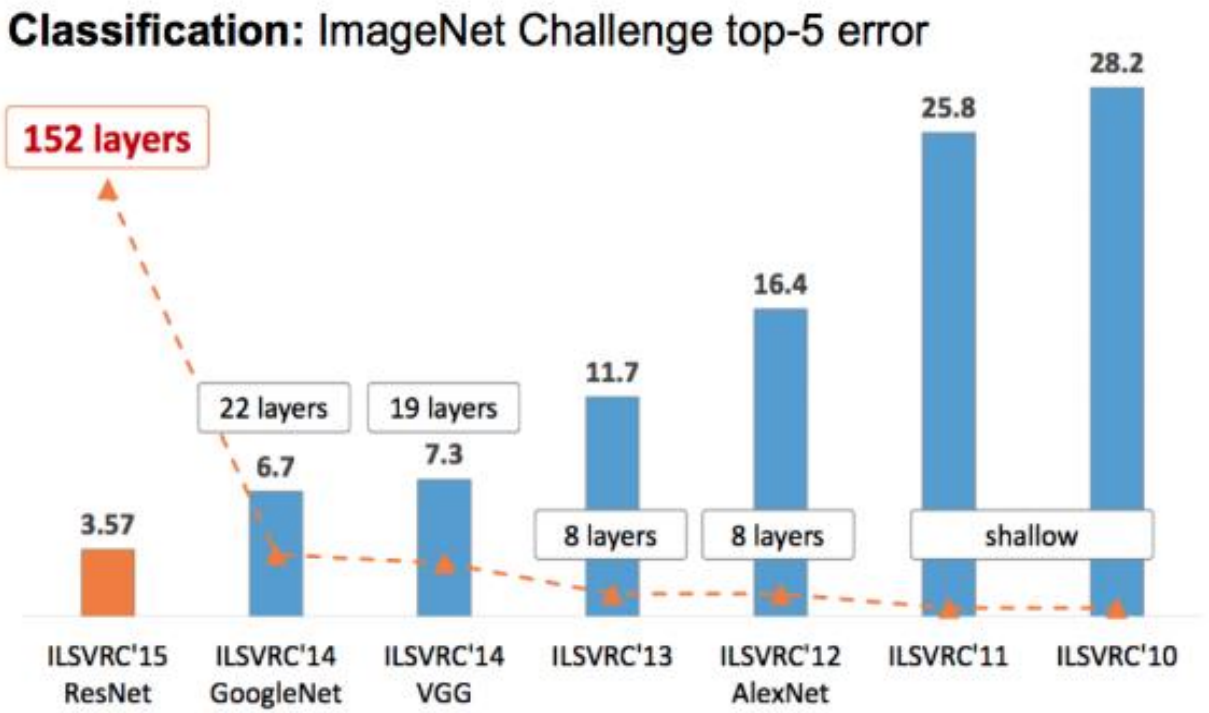
A table of Contents

Part 1

VGG net

Part 1

VGG net - 배경 & 정의



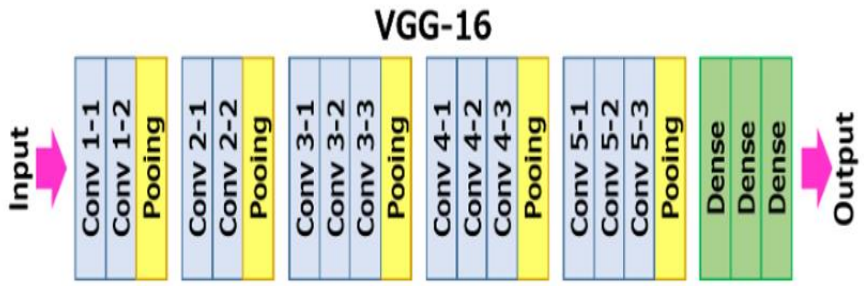
VGG net 배경

- 옥스퍼드 대학의 연구팀에 의해 개발
- 2014년 Image Net 이미지 인식 대회 준우승
- 사용하기 쉬운 구조와 좋은 성능으로 많은 인기

Part 1

VGG net - 배경 & 정의

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



VGG net 정의

- 'Visual Geometry Group Net'의 약자로
이미지 분류 예측 문제를 위한 모델

- 몇 개의 층으로 구성되어 있는지에 따라
VGG16, VGG19로 구분

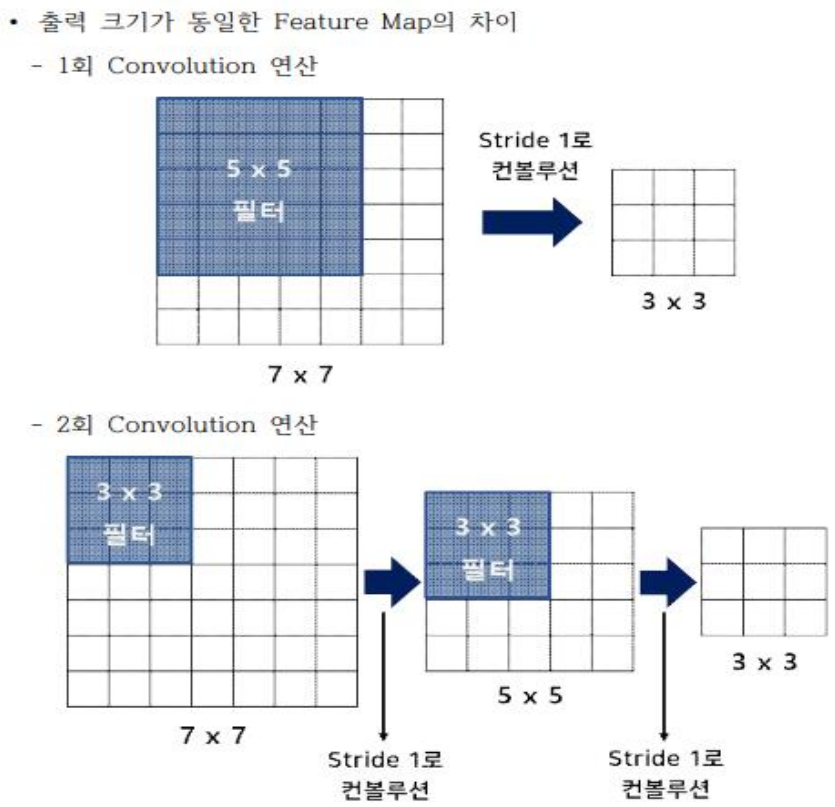
- D 구조가 VGG16, E 구조가 VGG19에 해당

softmax 함수로 활성화 된 출력값 → 1000개

제가 준비한 Data set의 클래스 개수에 맞게
완전연결층을 추가하여 사용!!

VGG net - 특징 : Structure 측면

[1] Factorizing Convolution



ReLu 함수를 더 많이 사용 → 비선형성 증가

이미지의 상,하,좌,우 정보를 반영할 수 있는
최소한의 Receptive Field

1x1 Convolution Filter → Spatial한 정보를 놓침

VGG net - 특징 : Structure 측면

[2] Pre-Initialization

Problem

Network가 깊어짐 → Vanishing/Exploding gradient 문제

Solution

11-layer VGG Net 먼저 학습

학습된 Parameter를 기반으로 모델에
Convolutional Layer를 추가하여 나머지 구조 완성

VGG net - 특징 : Structure 측면

[3] FC Layer → Conv. Layer

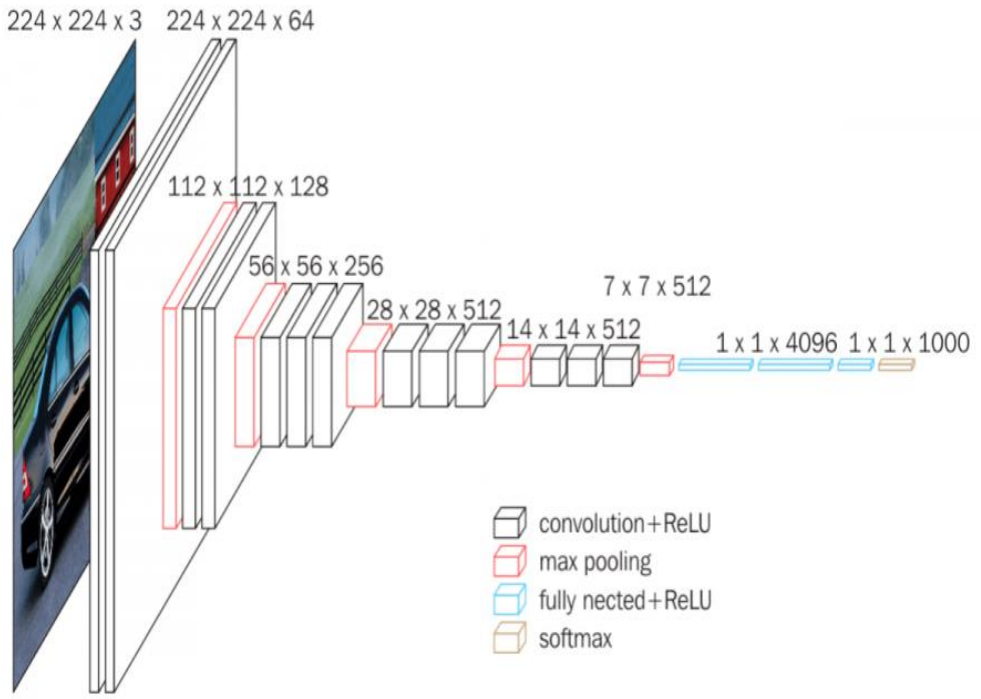


Image scale과 상관없이 Class를 구분하기 위해
FC layer를 Conv. Layer로 변경

Sum-pooled(Average) 작업을 통해
1x1 Feature map으로 변경하여 Class를 구분

VGG net에서는 첫 번째 FC Layer를 7x7 Conv. Layer,
나머지 2개의 FC Layer를 1x1 Conv. Layer로 변경

Part 1

VGG net - 특징 : Data set 측면

[1] Scale Jittering

Multi scale 방법으로 이미지의 scale을 256 ~ 512 범위에서 랜덤 선택

⇩

다양한 크기의 이미지에 대응

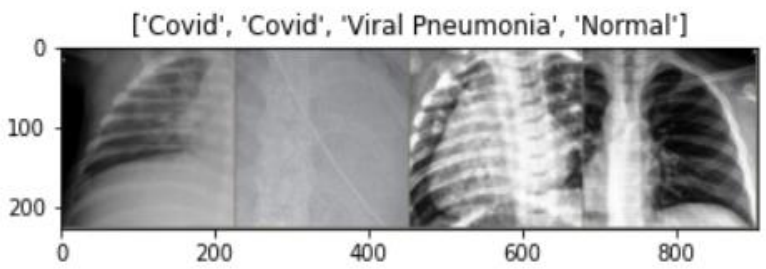
[2] Multi-Crop + Dense Evaluation

- **Multi-Crop:** 하나의 이미지 → 150장의 이미지 추출
- **Dense Evaluation:** Max pooling을 촘촘히 적용하여 표현력이 떨어지는 것을 방지

* ‘Multi-Crop evaluation’과 ‘Dense Evaluation’은 서로 상보성*

Part 1

My Data set



Covid-19 Data Set

Covid-19 data set:
https://www.kaggle.com/pranavraikokte/covid19-image-dataset

Train data: 251
Validation data: 66

Class(3): Normal, Covid, Viral Pneumonia

Food Data Set

Food data set:
https://www.kaggle.com/pcharith/fooddset

Train data: 4200
Validation data: 635

Class(6): butternaan, chicken, dhokla, dosa, samosa, vadapav

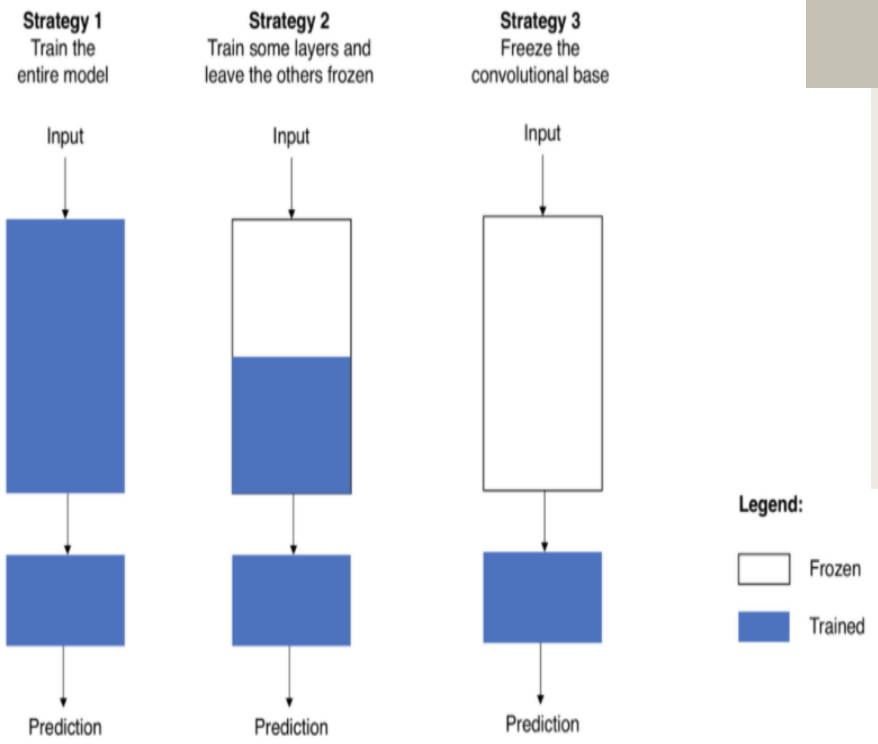
Cat_Dog Data Set

Cat_Dog data set:
https://www.kaggle.com/chetankv/dogs-cats-images

Train data: 8000
Validation data: 2000

Class(2): cats, dogs

전이 학습 : Transfer Learning



전이 학습

Image Net의 Data set으로 학습한 가중치의 일부를
능력이 유사한 새로운 신경망에 복사한 후, 재학습 수행

⇩

데이터의 크기가 작을 때 효과적 + 속도 빠름 + 정확도 상승

#1 – Parameter + 기본 구조만 그대로 사용 → 전체 새로 학습
#2 – Convolutional Base의 일부분 고정 → 나머지 계층과 Classifier 새로 학습
#3 – Convolutional Base는 고정 → Classifier만 새로 학습

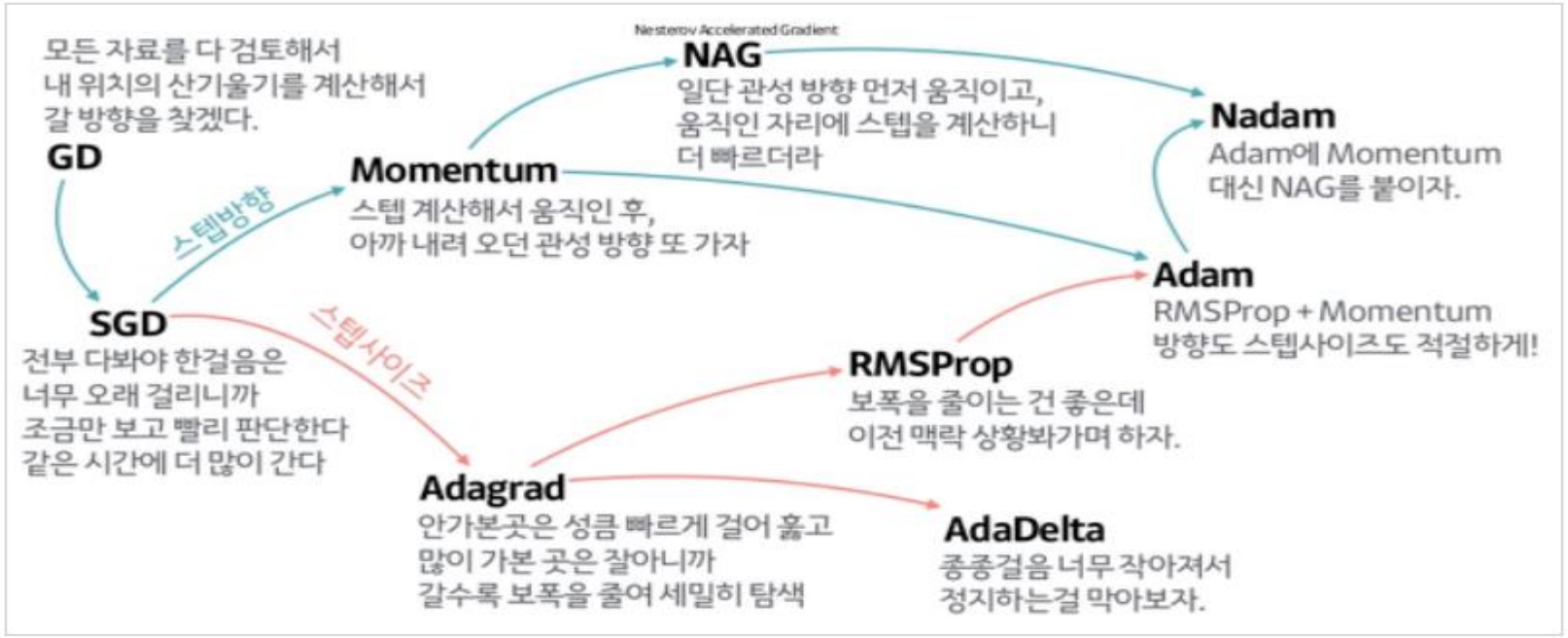
Part 2

Optimizer

Optimizer

정의: 신경망의 가중치와 절편을 학습하기 위한 알고리즘 또는 방법

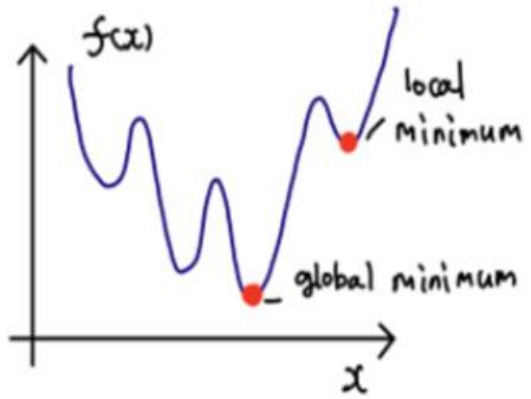
→ Network가 빠르고 정확하게 학습하는 것이 목표



Part 2

Optimizer - Momentum

Momentum



$$V_t = m \times V_{t-1} - \eta \nabla_\omega J(\omega_t)$$

$$\omega_{t+1} = \omega_t + V_t$$

SDG → 매우 느림 + global이 아닌 local minima에 수렴할 가능성 농후

빠른 학습 속도와 local minima에 빠지는 문제를 개선
→ 관성의 개념 적용

Vt는 이전 이동 거리와 관성 계수 m에 따라
Parameter Update

Part 2

Optimizer – RMSprop

RMSprop

Gt 계산식에 지수이동평균을 적용

Parameter 간의 차별화는 유지하고 학습속도가 지속적으로 줄어들어 0에 수렴하는 것은 방지

Decaying factor는 Hyperparameter로 0.9~0.999의 값

Adagrad와 같이 gradient 제곱으로 학습률을 나누지만

최근의 gradient를 사용하기 위해 지수 감소 사용

$$G_t = \gamma G_{t-1} + (1-\gamma)(\nabla_\omega J(\omega_t))^2$$

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla_\omega J(\omega_t)$$

Optimizer - Adam

Adam

앞에서 다룬 'RMSprop'과 'Momentum' 기법을 합친 옵티마이저

Adam에서는 기울기 값과 그 제곱값의 지수 이동평균을
함께 고려하여 Step 변화량을 조절

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\omega J(\omega_t)$$

$$V_t = \beta_2 m_{t-1} + (1-\beta_2)(\nabla_\omega J(\omega_t))^2$$

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{V_t + \epsilon}}$$

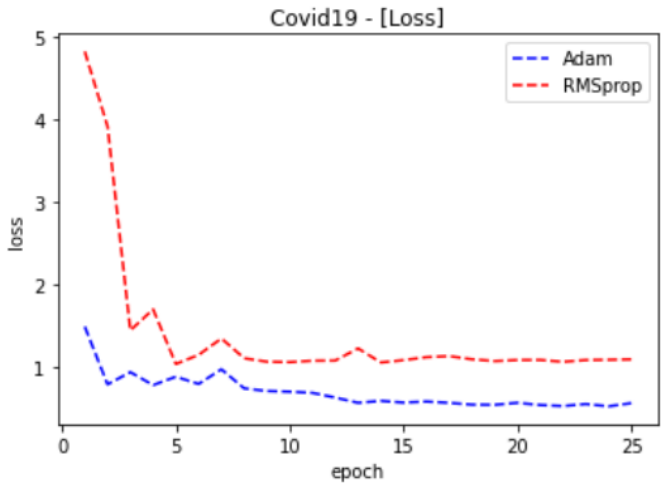
Hyperparameter는 각각 $\epsilon = 1^{-8}, \beta_2 = 0.999$의 값들이 추천된다.

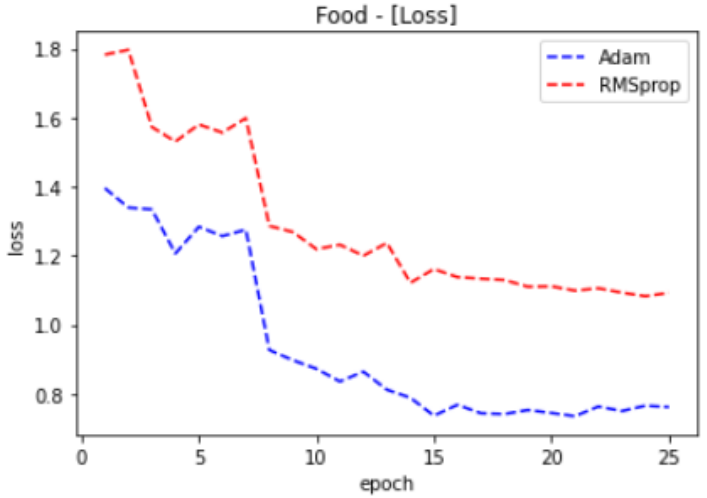
Part 3

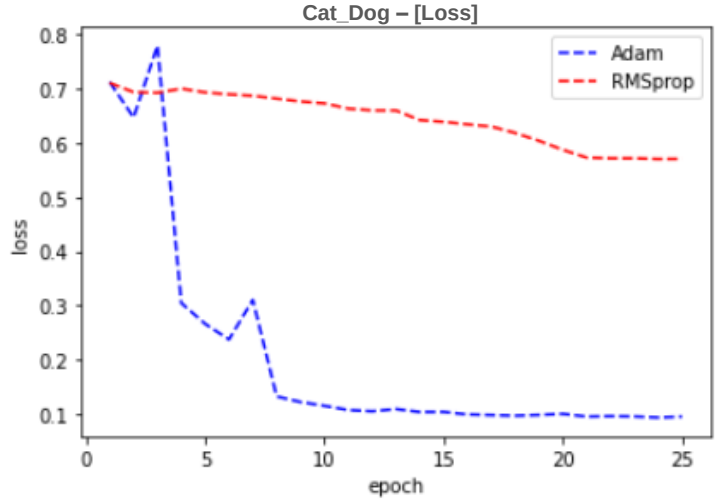
Results

Part 3

Results – Comparing 'Loss'

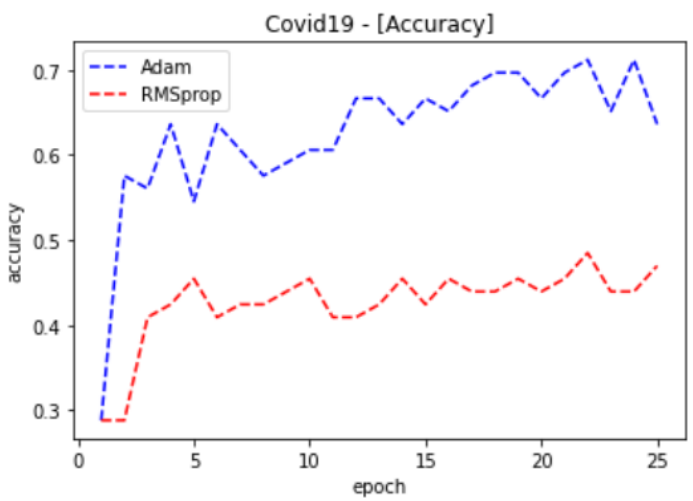


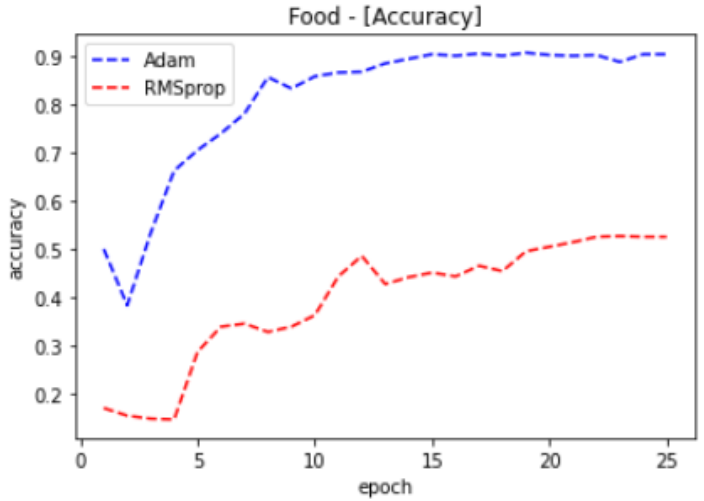


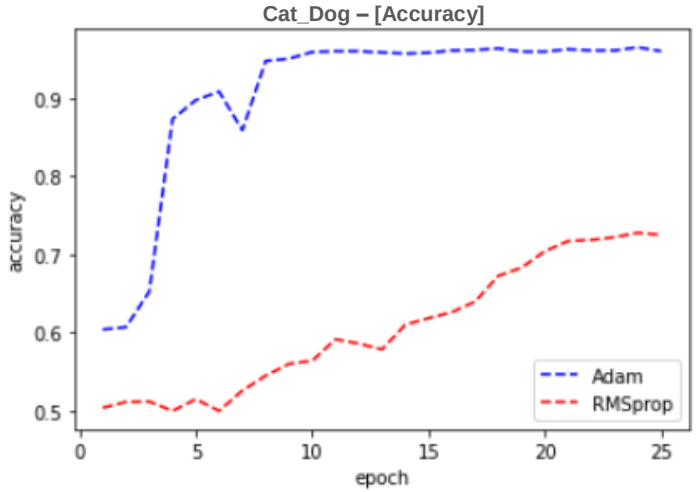


Part 3

Results – Comparing ‘Accuracy’







Reference

Websites & documents

[1]https://minjoos.tistory.com/6

[2]https://neocarus.tistory.com/entry/VGGNet-CNN-architecture-%EA%B8%B0%EB%B0%98-%EB%AA%A8%EB%8D%B8

[3]http://contents2.kocw.or.kr/KOCW/data/document/2020/edu1/bdu/hongseungwook1118/111.pdf

[4]https://oi.readthedocs.io/en/latest/computer_vision/cnn/vggnet.html

[5] https://welcome-to-dewy-world.tistory.com/92

[6] https://gaussian37.github.io/dl-dlai-RMSProp/

Alticles

[1] **K. Simonyan, A. Vedaldi, A. Zisserman,** "Deep Inside Convolution Networks: Visualising Image Classification Models and Sailency Maps", **Workshop at International Conference on Learning Representations, 2014**

[2] ***Aravindh Mahendran, Andrea Vedaldi, "***Understanding Deep Image Representations by Inverting Them", **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015**

감사합니다