2019 오픈소스 프로젝트

KU 건국대학교 KONKUK UNIV.

프로젝트 최종 보고서

- Scene Sentiment Classifier -

3조

201411235 임환규

201511173 강인한

201511180 김배승

201714150 김동진

201714157 이종완

2019년 12월 22일

목차

1. 소개 3

1.1 문제 정의 3

1.2 개발 동기 3

2. 관련연구 5

2.1 VGG [1] 5

2.2 Resnet [2] 5

2.3 Flownet [3] 6

2.4 GLCM-CNN [4] 7

2.5 Inception [5] 7

2.6 사용한 오픈소스 명시 8

3. 시스템 아키텍쳐 연구 9

3.1 싱글 프레임 학습 9

3.2 멀티 프레임 학습 10

4. 실험 및 실험결과 13

4.1 학습데이터 13

4.2 싱글프레임 학습모델 실험 16

4.3 멀티프레임 학습모델 실험 19

5. 결과분석 21

5. 결론 22

6.1 잘한 점 (성과 및 의의) 22

6.2 한계점 (부족한 점) 22

6.3 추가하면 좋은 점(발전 가능성) 22

6. 사용방법 23

7. 참고문헌 26

1. 소개

1.1 문제 정의

Scene Sentiment Classifier는 영화의 한 특정 장면만 보고도 해당 영화가 어떤 장르에 속하는 영화인지 판별할 수 있는 분류기이다. 불필요하게 영화 영상 전체를 확인할 필요 없이, 영화의 특정 장면을 입력으로 넣으면 해당 장면을 추출한 영화가 어떤 장르에 속하는 영화인지 알려주는 기능을 수행한다.

입력은 영화의 특정 장면에서 추출한 두 장의 연속된 frame 이미지이며, 출력은 총 6개의 영화 장르 라벨이다. 장르는 액션, 공포, 2D 애니메이션, 3D 애니메이션, 우주 SF, 로맨스 영화로 구성된다. 학습에 필요한 영화 프레임 데이터는 직접 수집하였다.

각 장르별 8개씩, 총 48개의 영화 데이터로 학습을 진행하였으며, 각 영화에서는 약 600프레임씩 추출하였다. 정확도 검증을 위한 테스트 데이터는 학습용 데이터로 사용된 영화들과 별개의 새로운 영화를 각 장르당 1개씩 선별하였다. 테스트 데이터도 학습용과 마찬가지로 한 영화당 약 600프레임을 추출하여 사용하였다.

1.2 개발 동기

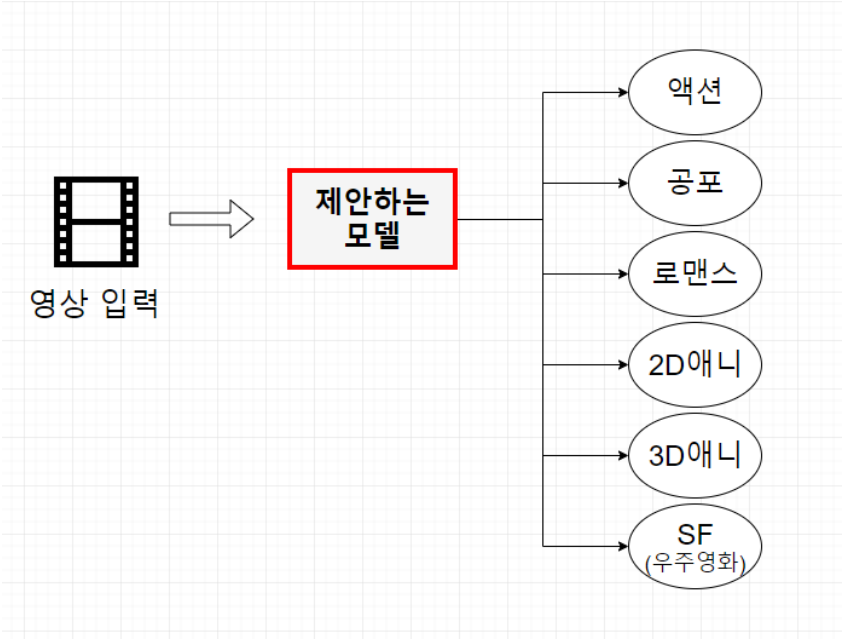
영화 장르 분류기를 만들고자 했던 계기는 다음과 같다.

첫번째로, 영화 관련 데이터베이스를 구축하는데 있어 사람이 수동으로 영화 장르 데이터를 라벨링 하는 작업은 매우 소모적인 일이다. 따라서 우리는 이러한 카테고라이징 작업을 자동화하여 불필요한 노동력 낭비를 줄이고자 한다.

두번째로, 개인 환경에 따라 사람이 임의의 기준으로 영화 장르를 구별한 결과는 부정확할 수 있다. 그러나, 학습된 신경망 모델을 통해 객관적인 관점에서 장르를 판별함으로써 장르 분류의 신뢰도를 개선시킬 수 있다고 생각한다.

마지막으로, 영상의 단일 프레임 데이터만을 사용하는 것이 아니라 다수의 프레임 사이에서 모션 정보를 추출하여 주관적 개념인 장르를 신경망에 학습시키는 일은 매우 도전적인 과제이다. 이론적으로는 모션 정보가 영화 장르에 대해 더 정확하고 많은 정보를 담고 있을 것이라 생각하고, 이를 이용해 신경망을 학습하면 더 좋은 결과를 얻는 것이 가능하다고 생각한다. 하지만 모션 정보를 추출하고 신경망 학습에 적용시키는 것은 쉽지 않은 일이고, 실제로 정확도 개선을 위해

활용할 수 있을지도 확실하지 않다.



[그림 1.1] 영화 장르 식별기

1.3 결과 기대

1) 예상 결과

우리가 만들고자 하는 프로그램의 예상 제작 결과는, 영화의 한 장면에 해당하는 두 프레임 이미지를 입력으로 넣으면 해당 장면을 추출한 영화가 속하는 장르를 결과로 출력하는 영상 장르 분류 프로그램이다. 예상 모델의 입력 계층은 연속된 두 프레임을 그대로 6 채널로 쌓거나 모션을 추출하여 채널을 쌓은 형태이고, 출력 계층은 장르 수만큼의 노드로 이루어진다. 평가함수로는 크로스 엔트로피 함수를 사용한다.

2) 기대 효과

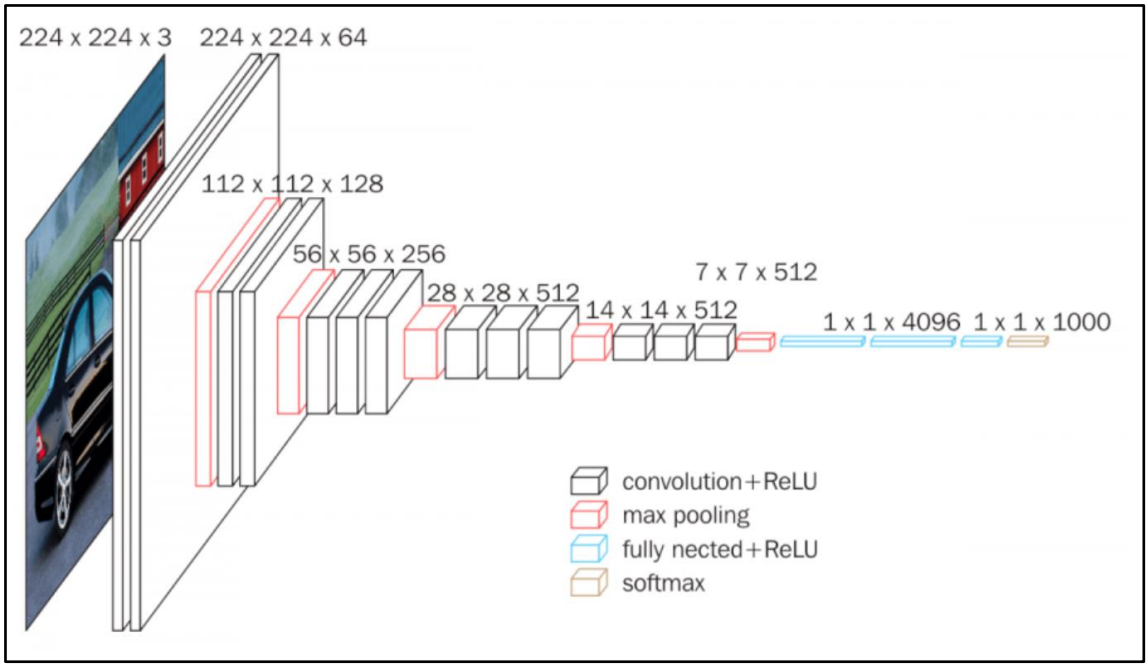
Scene Sentiment Classifier 를 사용하면, 사람이 직접 분류하지 않고도 영화의 장르를 객관적으로 분류할 수 있다. 이는 넷플릭스와 같은 영상 컨텐츠 제공 플랫폼에서 필수적인 카테고라이징 작업을 불필요한 노동력을 들이지 않고 수월하게 할 수 있음을 의미한다.

또한 유튜브 등 개인 미디어 컨텐츠 공유 플랫폼이 활성화됨에 따라 수많은 개인 영상들이 업로드 되고 있는데, 개인 업로드 영상들은 그 수가 너무 방대해 직접 카테고라이징 작업을 수행할 수 없고, 업로드하는 사용자 개인에게 맡기기엔 사용자에 따라 라벨링이 주관적이거나 정확하지 않을 수 있고, 혹은 생략되기도 한다. 따라서 프로그램을 통해 영상 정보 만으로 영상이

속한 장르를 객관적으로 분류해낼 수 있다면 영상 플랫폼에서 필수적인 추천 알고리즘의 정확도를 개선시키는 데에 효과적으로 사용될 수 있을 것이다.

2. 관련연구

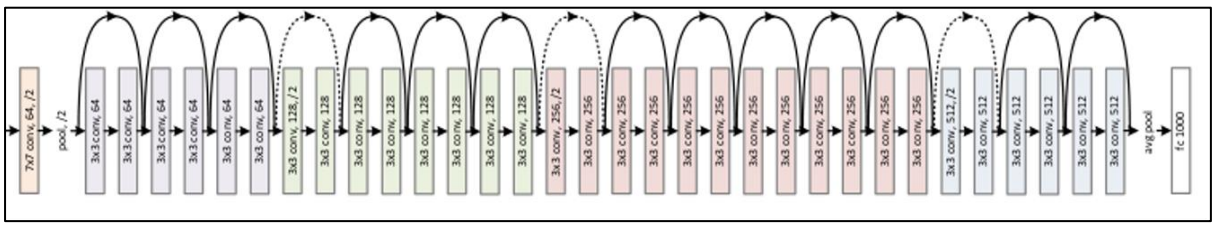
2.1 VGG [1]



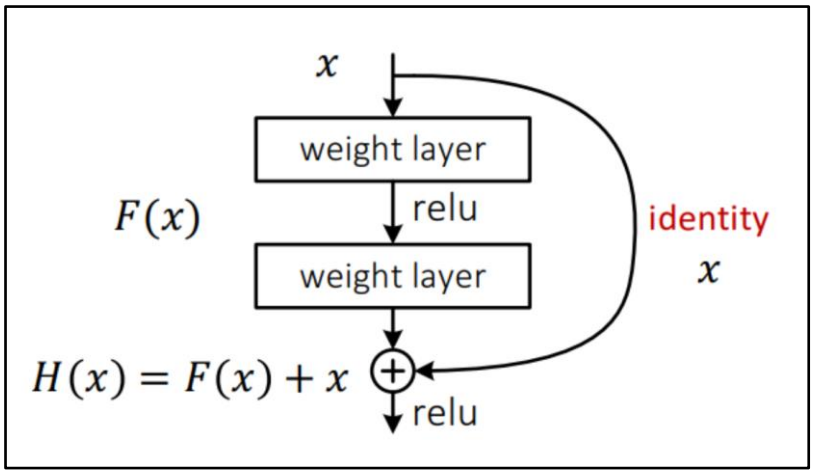
[그림2.1] VGG16 구조도

VGG 네트워크의 핵심은 신경망의 깊이이다. 그렇기 때문에 모든 convolution 계층의 필터의 크기는 3x3이고, max pooling 계층이 중간중간 마다 삽입되어 있다. 입력 데이터의 크기는 224X224x3 이고, 신경망을 들어 갈수록 점차적으로 데이터의 가로 세로 크기는 감소한다. 마지막 3 계층은 fully connected 계층이다. 설계상 출력 데이터의 크기는 1x1x1000 즉, 1000개의 클래스로 분류가 가능한 모델이다.

2.2 Resnet [2]



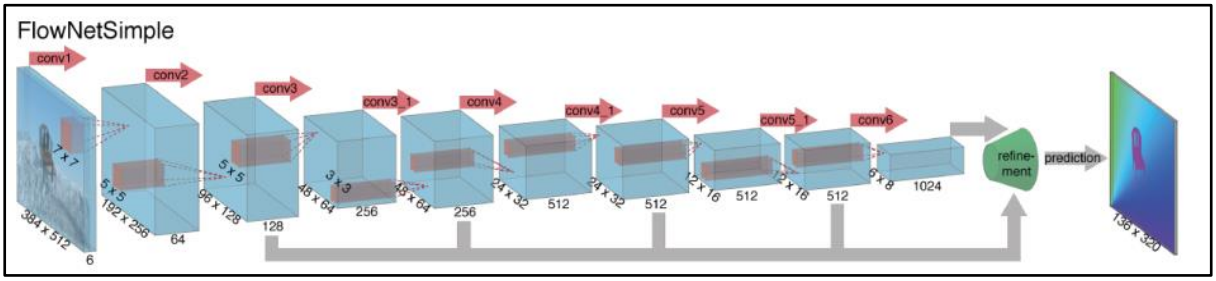
[그림2.2] Resnet34 구조도



[그림2.3] Residual block 구조도

Resnet의 핵심은 그림3에서 제시되었듯이 Residual block이다. Normalization 또는, weight decay를 적용시켰을 때, 신경망이 깊어 질수록 학습이 잘 안되는 현상이 발생하였다. 그 이유는 신경망이 깊어지면 매개변수들의 정보들이 사라지기 때문이다. 따라서, 신경망이 깊어져도 온전한 학습을 진행하기 위하여 Residual block 이 고안되었다. 위 그림에서 신경망을 온전히 거치고 나온 값인 F(x) 가 0으로 수렴해도, x의 값이 보존되어 덧셈 연산을 하기 때문에 정보 유실이 발생하지 않는다. 결론적으로, VGG 모델 구조에서 Residual block 개념을 적용 시킨 모델이 Resnet이다.

2.3 Flownet [3]



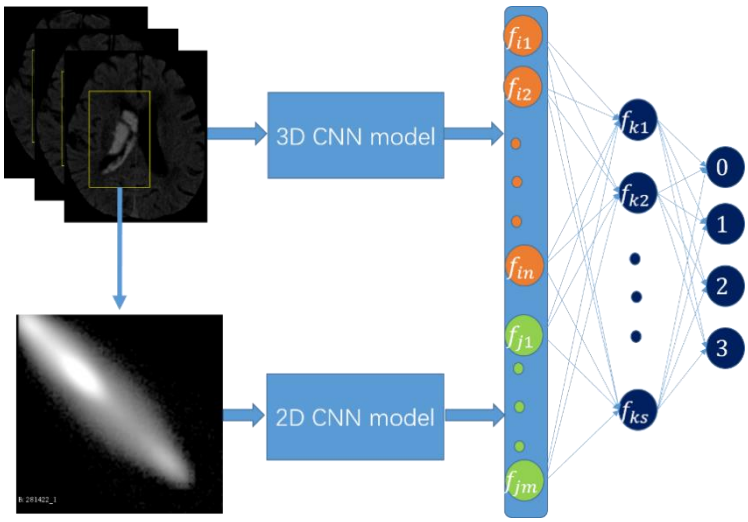
[그림2.4] Flownet-S 구조도

Flownet은 모션벡터 레이블을 갖는 연속된 프레임을 학습시켜서 optical flow estimation을 수행하는 딥러닝 모델이다. Optical flow는 픽셀 단위의 모션벡터의 집합을 의미한다. Flownet연구는 연속된 프레임의 간단한 학습 만으로 모션특징을 볼 수 있다는 점을 보여준다.

영화 장르 식별기에서 모션을 고려한 학습을 함께 연구할 계획이다. 그렇기 때문에 모션 특징을

추출할 수 있는 가장 간단한 모델인 Flownet-S 모델을 사용하여 실험을 진행하였다.

2.4 GLCM-CNN [4]

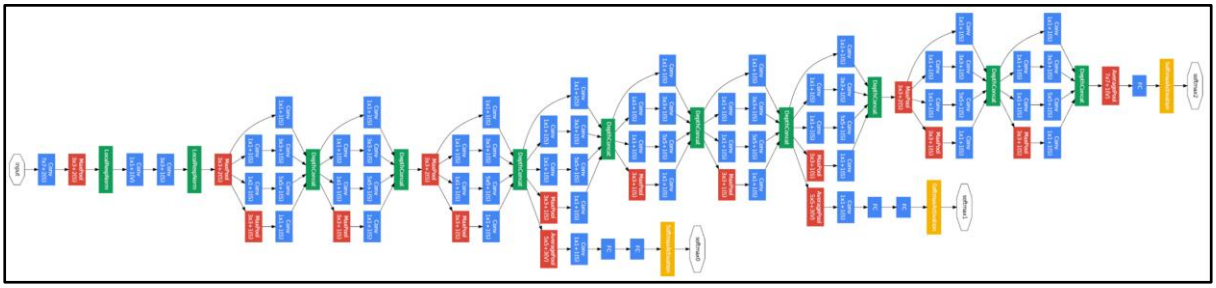


[그림2.5] GLCM-CNN 구조도

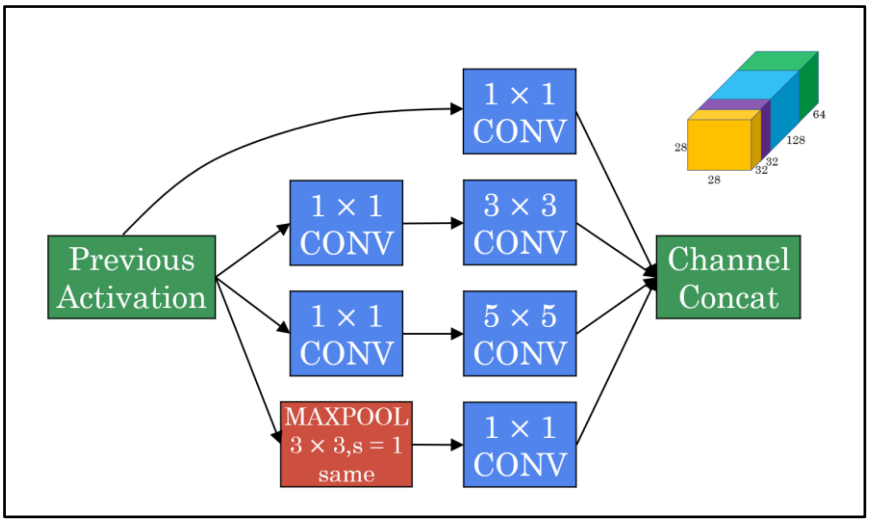
Hand-Craft Feature로 학습한 레이어와 Original Image를 학습한 레이어의 stack모델의 예로 GLCM-CNN 연구의 아이디어를 사용한다. GLCM은 의학 영상(CAD)에서 중요한 특징이지만, 딥러닝 모델이 학습을 통해 반영하기 어려운 특징이다. 그래서 GLCM에 Attention을 주기 위해 GLCM을 hand-craft feature로 추출한 후 학습하는 방법을 사용한다.

영화장르 분류기에서도 모션 정보를 사용하지만, 모션정보를 딥러닝 모델이 반영하기엔 Flownet에 한계가 있을 수 있다. 그러므로 GLCM-CNN과 같은 개념의 MOTION-CNN을 연구하였다. Auto-generation에 한계가 있을 수 있는 motion feature를 hand-craft feature로 학습하여 입력 영상과 Stack Network를 이루는 모델이다.

2.5 Inception [5]



[그림2.6] Inception 구조도



[그림2.7] Inception Module 구조도

Inception 네트워크의 핵심은 '다양한 convolution 필터 또는 max pooling 계층을 취사 선택하여 적용하여 보자' 이다. 위의 그림 7을 보면, 총 4가지 종류의 필터 또는 pooling 계층을 한 모듈에서 모두다 적용시키고 있다. 1x1 convolution 계층을 통하여 입력 데이터의 채널 수 조절이 가능합니다. 즉, 학습 연산량을 최적화 시킬 수 있다. 학습이 진행함에 따라 모델 매개변수에 의해 4가지 필터를 적절히 선택한 결과를 쌓아 올리면 Module의 출력 데이터가 된다. 결론적으로, Inception 모델은 앞서 설명한 Inception module들을 쌓여 올려서 만든 네트워크이다.

2.6 사용한 오픈소스 명시

Pytorch, OpenCV에서 제공하는 이미지 처리 기능을 사용하였으며, 네트워크 구현에 있어서 Inception v3 부분은 오픈소스를 이용하였다.

3. 시스템 아키텍쳐 연구

3.1 싱글 프레임 학습

싱글 프레임 학습은 단순히 R,G,B 색상 데이터의 분포와 특징 만으로 classification을 수행하게 된다. 따라서, 신경망 학습에 있어서 입력데이터는 각 장르의 고유한 특징을 소유하고 있어야 된다. 또한, 입력데이터의 복잡한 특징을 명확히 잡아내기 위하여 많은 신경망 계층이 요구된다.

- **Full Connected Layers**

완전 연결 층만을 이용하여 만든 신경망 모델이다. 입력데이터는 120x68x3 의 크기를 갖는다. 단일 프레임을 입력데이터로 사용하므로 3개의 채널을 갖는다. 원본 데이터의 크기에 비하여 많은 부분 resize가 이루어지므로, resize 과정에서 중요한 특징을 유실할 가능성이 존재한다. 은닉층은 각각 50개, 10개, 40개로 3개의 완전 연결 층을 갖고, 6개의 클래스로 분류한다. 입력 데이터의 복잡한 특징을 잡아내기에는 부족한 은닉 계층을 가지고 있다. 또한, 신경망을 확장하기에도 완전 연결 층 특성상, 많은 량의 파라미터 수가 요구된다. 따라서, 신경망 구조 특성상 적절한 학습 모델이 아니다.

- **VGG**

VGG16을 이용하여 영화 장르를 식별하는 모델을 설계하였다. 기존 모델의 구조를 보았을 때, Full Connected Layers 모델 보다 많은 성능 향상이 기대된다. 복수의 CNN 필터 계층을 사용함으로써, 신경망이 깊어 짐에 따라 발생하는 연산량 증폭 문제를 해결 할 수 있다. 입력 데이터는 224x224x3 을 요구하며, 이 또한 적절한 이미지 크기로 resize 함으로써 특징 유실을 어느정도 방지할 수 있다. 또한 비교적 단순한 모델 구조 덕분에 신경망 구현과 응용이 원활하다.

- **Resnet**

Resnet 과 VGG의 모델의 차이는 Residual block의 적용 유무이다. 따라서, Resnet 모델을 통하여 학습 속도 향상을 기대 할 수 있다. 또한, 데이터 정규화와 신경망의 많은 계층으로 인한 vanishing gradient 문제도 해결 할 수 있다. 따라서 Resnet을 이용하여 영화 장르를 학습하는 실험을 진행하였다. 입력 데이터는 VGG와 동일하게 전처리하였다.

- **Inception v3**

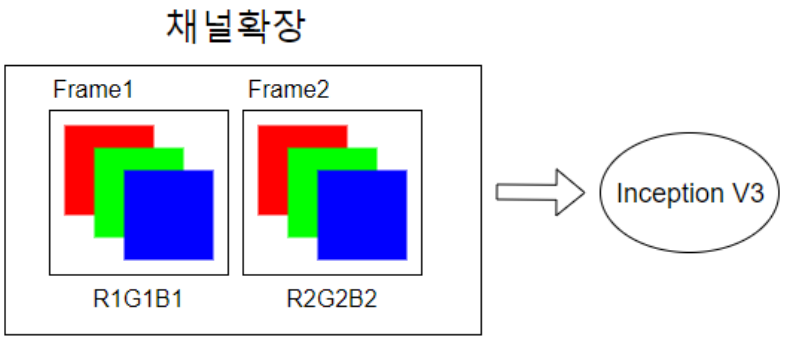
Inception v3 모델은 초기 Inception 모델이 가졌던 문제점들을 개선을 하였다. 구조 개선을 통하여 representational bottleneck과 factorization 문제를 해결하였다. representational bottleneck이란, 신경망 층 사이에서 이미지의 크기를 과도하게 줄이면, 정보 유실이 발생하는 문제를 의미한다. factorization 이란, convolution 필터의 크기를 작은 필터로 대체함으로써 연산량을 줄일 수 있음을 의미한다. 구조적 개선점을 통하여 입력 데이터를 학습시키는 데 있어서, 앞서 소개 되었던 모델보다 향상된 학습 결과가 예상된다. 입력 데이터는 단일 프레임 이미지인 299x299x3을 요구한다. 이는 VGG와 Resnet 모델 보다 더 큰 이미지를 처리할 수 있다.

3.2 멀티 프레임 학습

싱글 프레임 학습은 색조, 채도, gradient등을 고려하여 classification을 수행한다. 하지만, 영화 장르를 구분하는데 있어서 가장 중요한 특징은 motion이다. 액션영화는 대체로 모션이 역동적이고, 로맨스나 공포영화는 모션 량이 적기 때문이다. 따라서 멀티프레임을 학습하면 더 좋은 장르 식별기가 나올 수 있다. 멀티 프레임 학습을 위한 백본 아키텍처는 싱글 프레임 학습에서 가장 좋은 성능을 낸 Inception v3[5] 모델을 사용한다.

1) 채널 확장 모델

싱글 프레임 학습의 입력영상은 일반적으로 RGB채널을 사용한다. 멀티 프레임을 학습에 반영하는 가장 좋은 방법은 연속된 영상 두개의 채널을 쌓는 것이다. Flownet-S[3] 아키텍처는 멀티 프레임의 채널 확장 만으로도 CNN이 모션 정보를 학습할 수 있다는 사실을 입증한다. 이 사실을 이용하여, 영상 1과 영상 2의 채널을 쌓은 형태인 $R_1G_1B_1R_2G_2B_2$ 채널로 확장하여 학습한다.



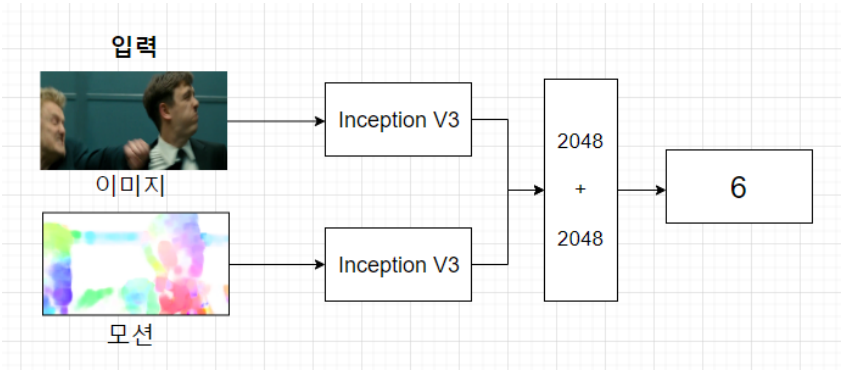
[그림 3.2.1] 멀티프레임을 이용한 입력 채널 확장

제안한 모델의 입력은 [그림 3.2.1]과 같다. 나머지 학습 과정은 싱글 프레임 학습에서 진행한 사항과 동일한 변수와 환경에서 진행하였다.2) 모션 피처(Feature) 스택(Stack) 모델

입력 채널 확장을 사용하여 모션 정보를 학습할 수 있지만, 모션추출을 CNN이 auto-generation 하는 비용이 크다. 그러므로 싱글 프레임 영상과 모션 정보를 각각 학습하여 stack하는 모듈을 사용한다. GLCM-CNN[4]을 참고하면 auto-generation이 어려운 특징을 따로 학습함으로써 실제로 중요한 특징에 attention을 줄 수 있다.

2-1) Double Stack Model

가장 단순한 아이디어는 이미지를 학습하여 얻은 뉴런과 모션을 학습하여 얻은 뉴런을 단순히 stack하는 방법이다. [그림 3.2.2]는 제안하는 double stack 방식의 네트워크이다. 그림에서 숫자로 적인 박스는 fc layer를 의미한다.

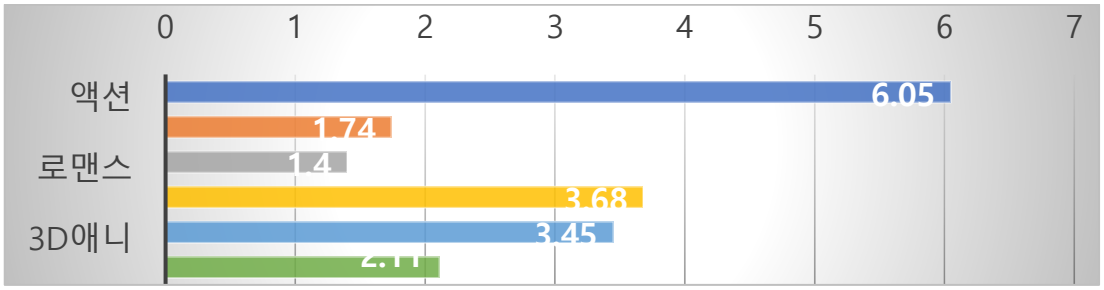


[그림 3.2.2] 더블 스택 모델

모션정보를 제외한 뉴런과 모션정보를 갖는 뉴런을 학습하여 4096 FC레이어를 한번 거친 후 출력 레이어에 도달한다. 파라미터 수가 많아 비효율적일수 있으나, 기존 모델에 모션 정보를 포함할 수 있는 직관적인 모델이다.

2-2) Motion Magnitude Stack Model

더블 스택 모델은 모션의 크기 뿐만 아니라 형태까지 반영한다. 하지만, 모션의 형태가 장르 식별에 좋은 영향을 줄지는 미지수이다. 하지만 [그림 3.2.3]을 보면 모션량(motion magnitude)은 장르 식별에 중요한 특징이라는 점을 알 수 있다. 모션량의 정의는 [식 3.2]이다.



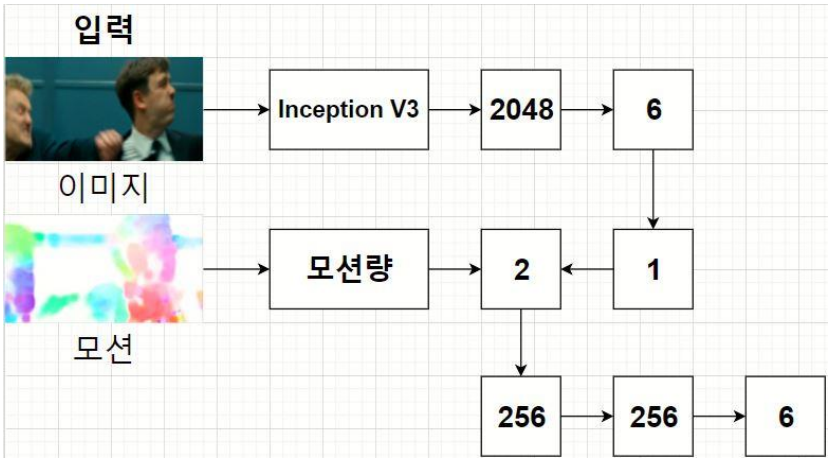
[그림 3.2.3] 모션량은 식별력이 높음

위의 결과를 토대로 모션량을 hand-craft하여 network stack을 진행한다. 아주 작은 네트워크의 추가로 연산량의 변화는 작지만 식별력 높은 특징을 부여할 수 있다는 장점이 있다.

$$모션량 = \sum_{k=0}^{n}\left(\frac{M_k}{H} * \frac{M_k}{W}\right)^{1/2} /N$$

[식 3.2] 모션량 정의, 이미지 사이즈(H,W), 이미지 개수(n), 모션 벡터의 크기(M)

추출한 모션량은 학습 이미지의 top fc layer를 하나의 뉴런으로 인코딩하여 사용한 뒤 정규화된 모션량을 추가한 뉴런을 256-fc 레이어에 걸쳐 최종 결과를 출력한다. [그림 3.2.4]는 보다 상세한 모델의 모습을 보여준다.



[그림 3.2.4] Motion Magnitude Stack Model

4. 실험 및 실험결과

4.1 학습데이터

1) 데이터셋 설명

학습을 위한 영화를 수집하기에 앞서 초기에는 7개의 장르(Action, Horror, Romance, Animation2D/3D, Musical)를 클래스로 선정했으나, Musical의 경우, 다른 클래스들과 고유 특징이 중첩될 가능성이 크다고 판단하여 제외시켰다.(역동적 움직임-Action과 겹침, 분위기-Romance와 겹침). 그러므로 최종 선정 장르는 다음과 같은 6가지 클래스이다.

Action, Horror, Romance, Animation(2D), Animation(3D), SF

각 장르 별 영화를 선정하고, 특징이 깊은 장면(프레임)들을 추출하여 데이터 셋을 구성했다. 아래의 표는 선정한 학습/테스트 데이터를 위한 영화들을 나타낸다.

장르 / 용도	Action	Horror	Romance	Ani_2D	Ani_3D	SF
TRAIN	스파이더맨	주온 1	어바웃타임	하울의 움직이는 성	드래곤 길들이기	퍼스트 맨
	분노의 질주	주온 2	500일의 썸머	짱구 극장판	인크래더블2	그래비티
	본 아이덴티티	착신아리	플림	너의 이름은	마이 펫의 이중생활	인터스텔라
	킹스맨	링	이프온리	목소리의 형태	주먹왕 랄프	라이프
	미션 임파서블	더 넌	러브 로지	라이언 킹	인사이드 아웃	가디언즈 오브갤럭시1
	테이큰	컨저링2	로맨틱 홀리데이	원피스	토이스토리	가디언즈 오브갤럭시2
	솔트	애나벨	조	건담	슈렉	스타트렉
	매드맥스	주온 3	타이타닉	나루토	라따뚜이	마션
TEST	어벤져스: 앤드 게임	셔터	러브 액츄얼리	코난 극장판	겨울 왕국	패신져스

[표 4.1] 데이터 셋을 위한 학습/테스트 영화

녹색 영화들은 중간 발표 이후 추가된 학습 데이터이다. 각 영화 당 총 600 프레임을 추출해서 약 600 * 54 = 32,400 프레임으로 구성된 데이터 셋을 만들었다. 학습용 데이터는 약 28,800 프레임, 테스트 데이터는 약 3,600 프레임을 가지고 있다.

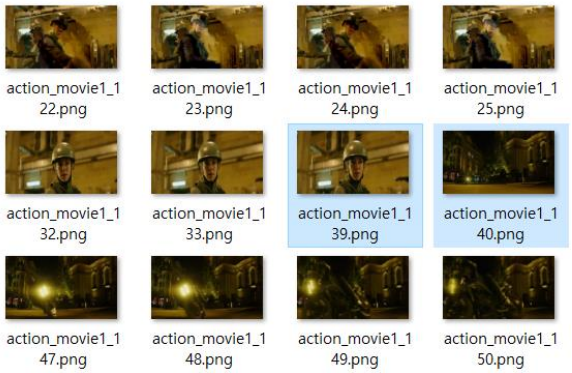
2) Single-frame 데이터 추출

Python으로 툴을 제작하여 직접 데이터 추출 및 라벨링을 진행했다. 유의미한 구간이라고 생각되어지는 부분의 시작과 끝을 hh:mm:ss 시간 단위로 입력 받는다. 입력 받은 시간을 프레임 번호로 변환 했을 때, 영상(영화) 내에서 유효한 프레임 구간으로 식별이 된다면, 해당 구간에서 랜덤으로 홀수 번호의 키 프레임들을 뽑는다. 이때 총 키 프레임은 영상 당 300 프레임을 선발한다. 그 후, 모션 정보를 얻기 위해, 키 프레임과 해당 키 프레임의 바로 다음 프레임(key+1)을 연속된 프레임 쌍으로 [그림 4.1.1]과 같이 저장한다. 즉, (key 프레임, key + 1 프레임) * 300 = 600 프레임이다.

3) Multi-frame 데이터 추출

2-1) 데이터셋 정제

기존의 단일 프레임 데이터를 추출할 때 추출기로 선별한 데이터셋을 그대로 사용하려고 했으나, 한 가지 문제점이 존재한다는 것을 알게 되었다. 기존 영화 데이터셋을 만드는 방식에서 비롯된 문제점이었는데, 바로 한 영화가 모두 같은 장면의 프레임들로 이루어지지는 않았다는 점이다.



[그림 4.1.1] action 장르의 train dataset 중 장면 전환이 일어나는 예시

위 [그림 4.1.1]에서 139-140 frame과 같이 장면 전환이 일어나는 부분이 문제가 되는 구간이다. 단일 프레임을 추출할 때는 문제가 되지 않았지만, 2장의 프레임을 쌍으로 추

출했을 때 간혹 장면 전환하는 부분의 프레임 2장이 같이 추출되는 문제가 발생하는 것을 확인했다.

이와 같은 문제점을 해결하기 위해 먼저 프레임 번호를 세 자리 숫자로 통일하고, 반드시 홀수+짝수 프레임 한 쌍으로 추출하는 방식으로 고정하였다. 그 다음, 홀수 번째 프레임과 짝수 번째 프레임이 서로 다른 장면이거나, 각각 다른 구간에서 추출된 프레임이라면 그 두 프레임을 모두 삭제한다. 이 방법으로 잘못된 데이터가 추출되는 문제를 방지할 수 있었다.

2-2) 연속된 2-frame 데이터 추출

학습데이터 입력을 다중 프레임으로 만들기 위한 구현을 하였다. 단일 프레임 추출 방식에서 이중 프레임 추출 방식으로 변경하기 위해 두 부분을 수정하였다.

첫번째로, 랜덤으로 뽑아진 index에 대해 실제 파일 경로를 받아와서 파일 이름에 넘버링 된 숫자를 확인한다. 그 후, 홀수+짝수 프레임 쌍으로 추출하기 위해서 프레임 넘버가 홀수이면 현재 프레임을 첫 번째 프레임으로, 다음 프레임을 두 번째 프레임으로 가져온다. 반대로 프레임 넘버가 짝수이면 현재 프레임을 두 번째 프레임으로, 이전 프레임을 첫 번째 프레임으로 가져온다.

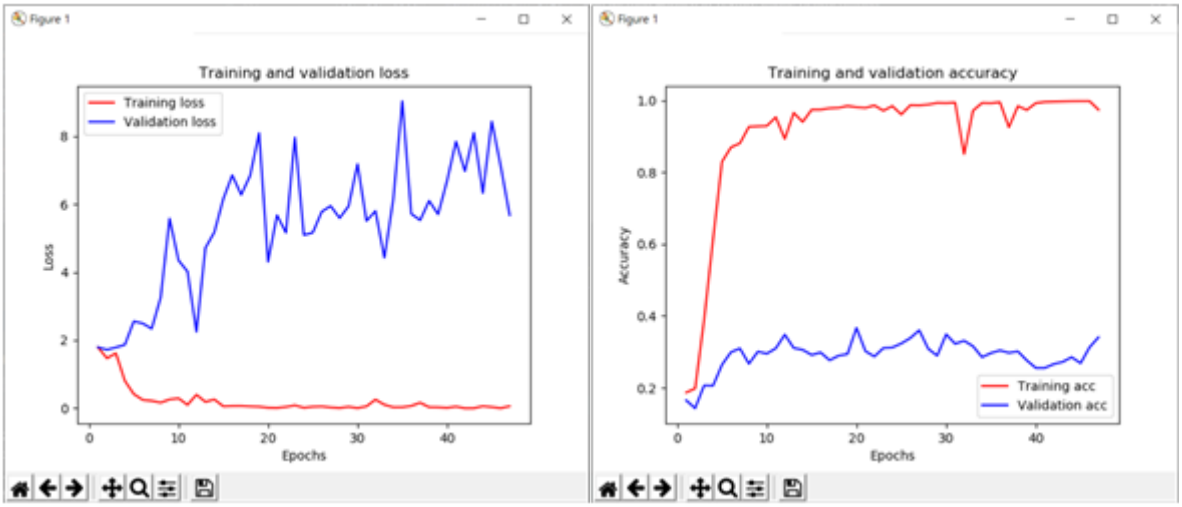
두번째로, 가져온 두 프레임에 대해 stacking 작업을 수행한다. RGB 3채널 이미지 2장을 쌓아 총 6채널의 RGBRGB 이미지를 생성한다. 이러한 방법으로 프레임 쌍을 추출하여 반환한다.

4.2 싱글프레임 학습모델 실험

앞서 제안한 네트워크를 학습하여 테스트한 실험 및 결과이다.

1) Fully Connected 네트워크

- 수업 과제를 베이스로 만든 네트워크
- Input Layer: flatten 120*68*3 이미지
- Hidden Layers
  - 1st Layer: 50 노드,
  - 2nd Layer: 10 노드,
  - 3rd Layer: 40 노드
- Output Layer : 6 노드(장르/클래스의 수)
- Epochs = 50, Dropout 적용 X
- Test Accuracy : 36.7%



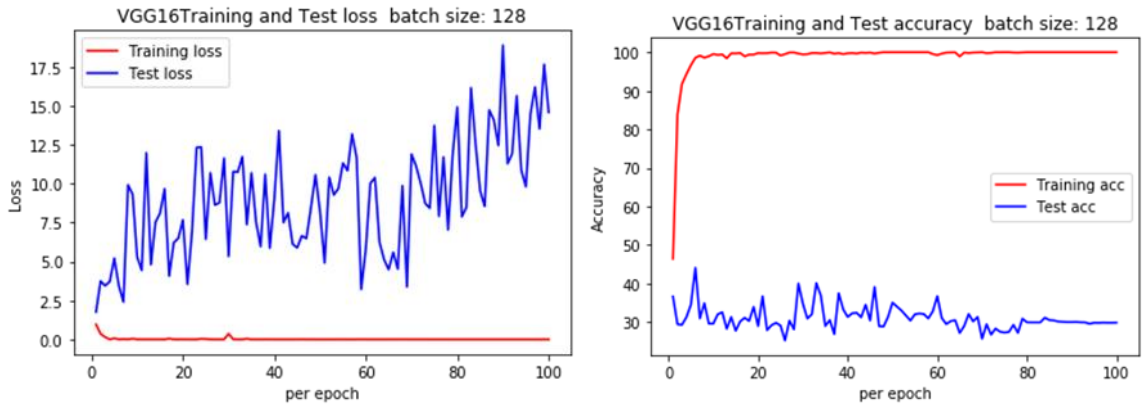
[그림 4.2.1] 이론 넷 loss & acc. 그래프

FC레이어는 공유 가중치를 사용하지 않기 때문에 공간적인 특징을 학습하기가 어렵기 때문에, 높지 않은 테스트 정확도가 나왔다.

2) VGG16

- Input Layer: resized 224*224*3 이미지
- Output Layer : 6 노드(장르/클래스의 수)

- Epochs = 100, Batch_size = 128, Optimizer = SGD(lr = 0.001, momentum = 0.9)
- Test Accuracy : 44.1%

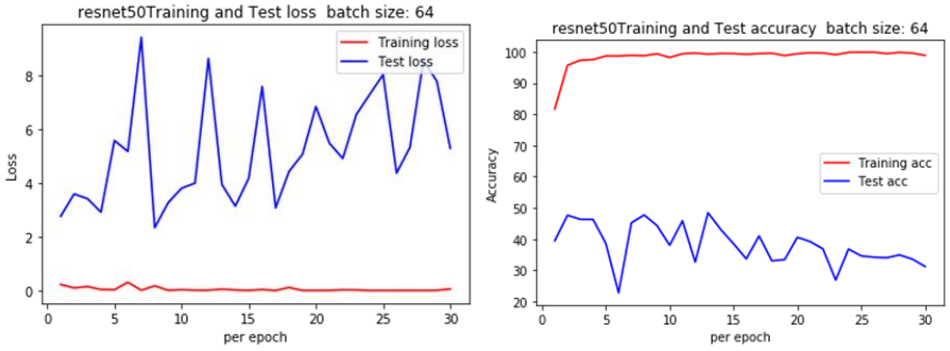


[그림 4.2.2] VGG16 loss & acc. 그래프

FC레이어보다 높은 결과가 나왔다. 하지만 VGG는 네트워크 깊이에 한계가 있기 때문에 주요한 특징을 추출하지 못하는 듯하다.

3) ResNet50

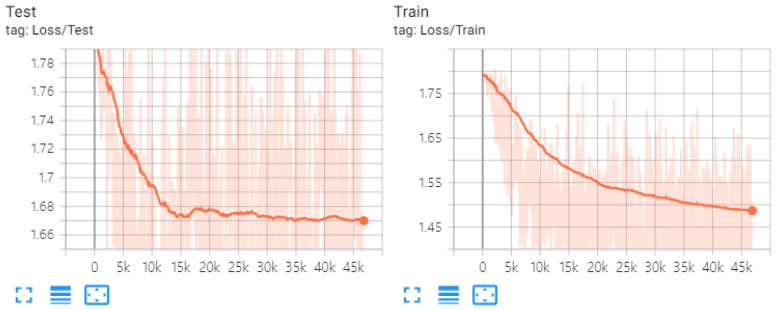
- Input Layer: resized 224*224*3 이미지
- Output Layer : 6 노드(장르/클래스의 수)
- Epochs = 30, Batch_size = 128, Optimizer = SGD(lr = 0.001, momentum = 0.9)
- Test Accuracy : 48.4%



[그림 4.2.3] ResNet50 loss & acc. 그래프

4) FlowNet + FC

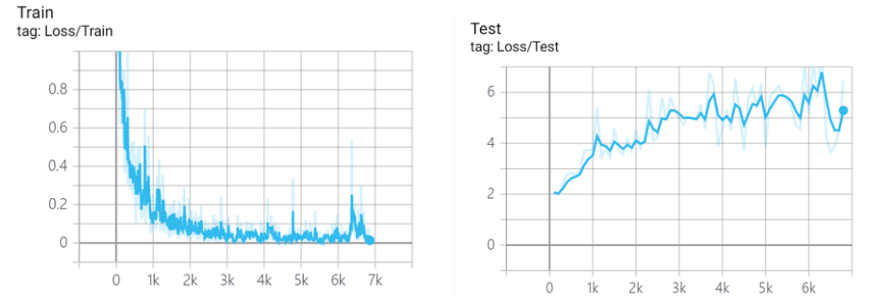
- FlowNet의 Classification 응용
- Dropout 적용, BatchNormalization 적용
- Epochs = 60, lr = 0.01, lr_Decay_Rate = 0.96, 초기 가중치 = (표준분포,분산 0.05)
- Test Accuracy : 51.8%



[그림 4.2.4] FlowNet+FC train loss & test loss 그래프

5) Inception v3

- Input Layer: resized 224*224*3 이미지
- Output Layer : 6 노드(장르/클래스의 수)
- Epochs = 20, Batch_size = 128, Optimizer = Adam(lr = 0.0001, momentum = 0.9)
- Test Accuracy : 59.2%



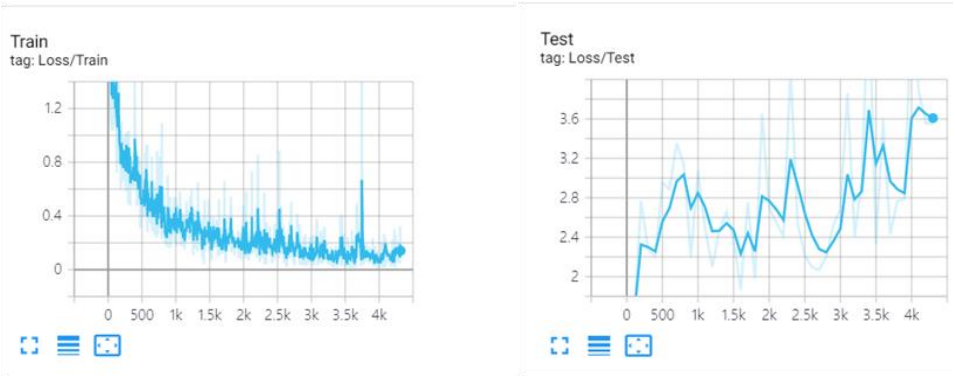
[그림 4.2.5] Inception_v3 train & test loss 그래프

4.3 멀티프레임 학습모델 실험

본 연구는 단일 프레임의 학습은 모션을 반영할 수 없기 때문에 다중 프레임을 가공하여 학습하는 방법을 제안하였다. 앞서 제안한 다중 프레임 학습 방식에 대한 효과성을 실험하였다.

1) 6 channel 학습

- Input Layer: resized 299*299*6 이미지
- Output Layer : 6 노드(장르/클래스의 수)
- Epochs = 20, Batch_size = 64, Optimizer = Adam(lr = 0.0001, momentum = 0.9)
- Test Accuracy : 53.8%



[그림 4.3.1] 6 channel learning train & test loss 그래프

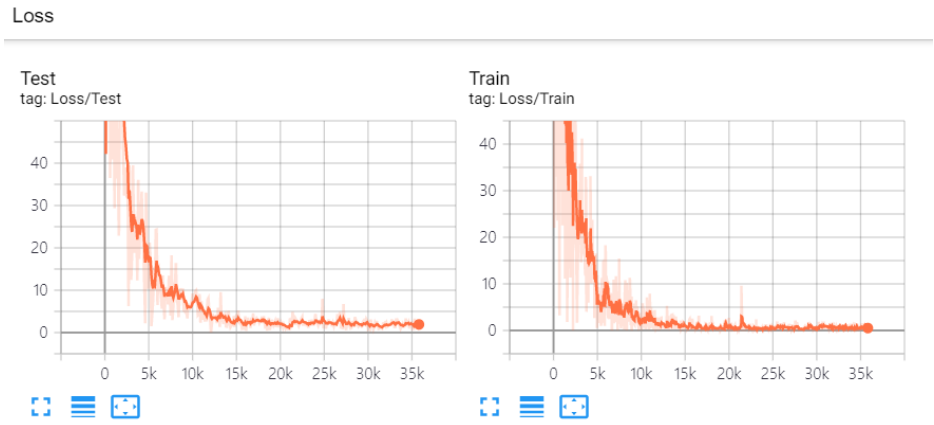
2) Double Stack Model

- Input Layer: resized 299*299*6 이미지
- Output Layer : 6 노드(장르/클래스의 수)
- Epochs = 50, Batch_size = 64, Optimizer = Adam(lr = 0.0001, momentum = 0.9)
- Test Accuracy : 52.7%

[그림 4.3.2] Double Stack Model Loss 그래프

3) Motion Magnitude Stack Model

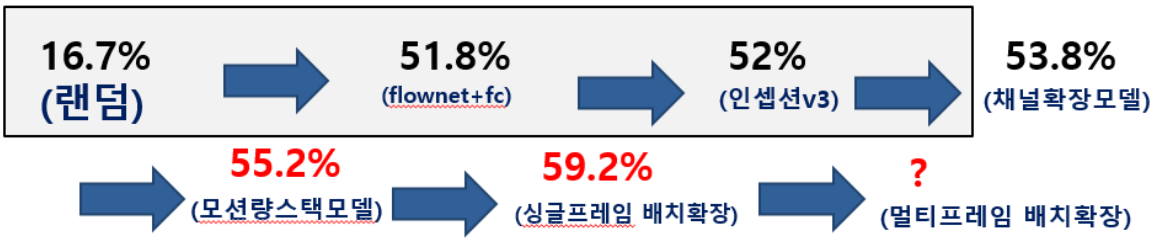
- Input Layer: resized 299*299*6 이미지
- Output Layer : 6 노드(장르/클래스의 수)
- Epochs = 50, Batch_size = 64, Optimizer = Adam(lr = 0.0001, momentum = 0.9)
- Test Accuracy : 55.2%



[그림 4.3.3] Motion Magnitude Stack Model Loss 그래프

5. 결과분석

네트워크 학습을 통해 단일프레임에서 52%의 정확도, 다중프레임에서 55.2%를 보였다. 학습데이터의 한계로 인한 data inconsistency의 발생으로 학습이 미니배치가 얼마나 질 좋은 데이터셋을 선별하는지에 따라 loss가 결정되는 문제가 있다. 그래서 학습 모델의 배치사이즈에 따라 데이터의 고른 선별이 이루어지면서 정확도가 올라가는 현상이 있다. 그래서 단일 프레임 학습을 배치 128로 확장한 결과가 59.2%이다. 충분한 메모리를 갖는 GPU에선 다중 프레임을 높은 배치사이즈에서 학습시킬 수 있으며, 기존 결과를 넘는 정확도가 나타날 것으로 기대한다. [그림 5.1]



[그림 5.1] 제안하는 모델 별 테스트 정확도

[그림 5.2] 테스트 어플리케이션 결과

5. 결론

6.1 잘한 점 (성과 및 의의)

본 프로젝트는 해당 문제에 대한 기반 연구 없이 진행하였다. 학습데이터 수집, 모델 설계 및 학습을 직접 수행하였으며, 의미 있는 정확도를 갖는 모델과 어플리케이션을 제작하였다. 뿐만 아니라 추가적인 모델 개선 방안을 제안함으로써 모션 정보가 영화의 장르를 결정짓는데 중요한 역할을 한다는 사실을 입증하였다.

6.2 한계점 (부족한 점)

- 2D : **95.8%**
- SF : **92.3%**
- 3D : **89.2%**
- 액션 : **41%**
- 공포 : **16%**
- 로맨스 : **26%**

[표 5.1] 장르 별 테스트 정확도 (3600 dataset)

[표 5.1]을 보면 정확도가 특정 클래스에 치우친다. 특히 공포와 로맨스 클래스는 정확도가 현저히 낮다. 이는 우리가 수집한 학습데이터가 영화의 주요한 특징을 아직 일반화할 수 없음을 의미한다. 또한 학습데이터가 모두 서양영화와 일본 애니메이션에 치중된 점을 미루어 보아 모든 영화에 강력한 모델은 아니다.

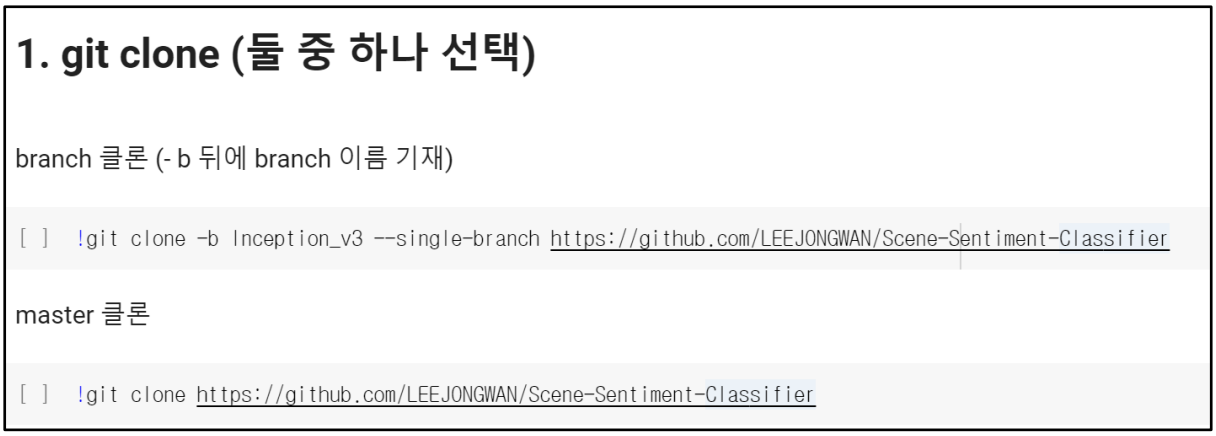
6.3 추가하면 좋은 점(발전 가능성)

더 많은 영화에서 무수히 많은 장면을 추출한 학습데이터 추가가 필요하다. 그러면 보다 일반화된 영화의 특징이 추출될 것이며 정확도가 상승할 것이다. 또한 모션을 학습하기 좋은 모델에 대한 추가적인 연구가 필요하다. 본 연구는 모션의 크기만을 사용해 성능을 높였다는 점에서 좋은 특징을 단편적으로 사용했다고 볼 수 있다. 효과적인 모델에 대한 추가적인 연구를 하면 모션을 추가한 모델에서 더 좋은 결과를 얻어낼 것이다.

6. 사용방법

우리 조의 모델 학습은 기본적으로 Google 사의 Colab 환경 위에서 진행되었다. 따라서, Colab 환경에서 소스코드 실행을 권장한다.

A. 소스 파일 중 ipynb 확장자를 가진 파일을 Colab에서 불러온다.



B. 파일을 성공적으로 불러오면 아래와 같은 셀 화면이 출력된다. 학습데이터 및 소스코드를 가져오기 위해서, git clone 과정이 필요하다. 두번째 셀인 'master 클론'을 선택하여 실행하면, 가상환경 저장소에 데이터 clone이 진행된다.

C. 클론이 성공적으로 완료되었으면, 실행 환경을 설정하기 위해 설치된 파일 내부로 디렉토리를 이동한다.

2.텐서 보드 설치

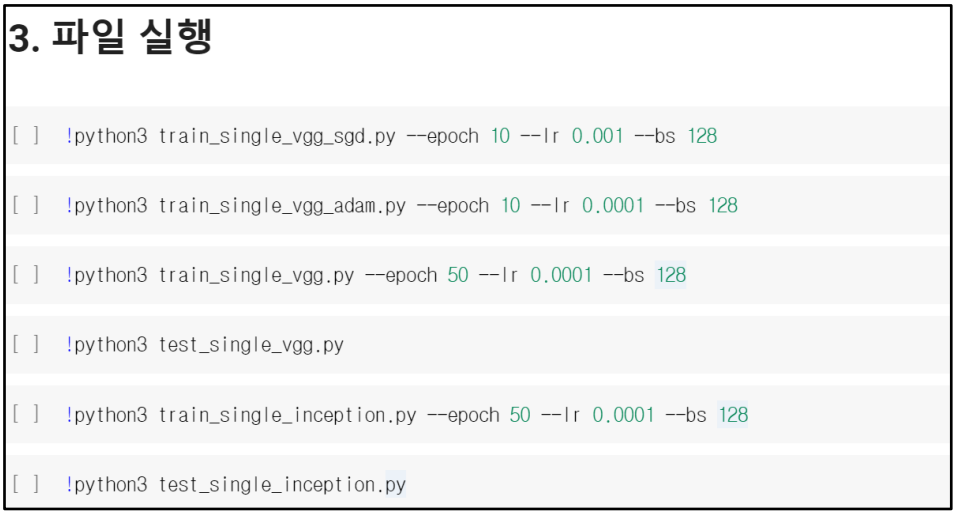
```
[ ]  !pip uninstall tensorboard
     !pip install --force-reinstall tf-nightly-2.0-preview

[ ]  !pip install --upgrade grpcio

[ ]  !pip install -q tf-nightly-2.0-preview

[ ]  %load_ext tensorboard

[ ]  import os
     logs_base_dir = "/content/Scene-Sentiment-Classifier/runs"
     os.makedirs(logs_base_dir, exist_ok=True)
```

D. 학습 및 테스트를 진행하기 앞서 우리 조는 텐서 보드를 이용한다. 텐서 보드란, 학습 진행 시, 다양한 정보들을 그래프로 출력해주는 프로그램이다. 따라서, 원활한 진행을 위해서 설치를 권장한다. 위의 그림과 같이 명시된 모든 셀을 실행하면, 텐서 보드 설치가 완료된다.

텐서 보드 실행

```
[ ]  %tensorboard --logdir {logs_base_dir}
```

E. 설치가 완료된 후, 텐서 보드를 실행하면 된다.



F. 모델 학습 소스 코드를 실행 시키기 위해서는 '!python3 + (파일 명) + (실행 매개변수)' 순으로 입력하면 된다. 이때, 실행 매개변수에는 총 3가지가 있다. Epoch는 모든 데이터를 학습 모델에 몇 번 넣을 것인가를 의미한다. lr 는 learning rate을 의미하며, optimizer가 모델 매개변수를 한번에 얼마나 조정할 값인가를 나타낸다. Bs는 batch size로 모델의 입력 데이터로 넣을 때, 몇 개의 데이터를 묶어서 넣을 것인지에 대한 수치이다. 이때, batch size는 GPU의 메모리 크기의 영향을 받는다. 큰 값을 넣을 경우, GPU 메모리 부족으로 원할 한 실행을 보장하지 않는다.

G. 모델 학습이 완료된 후, 동일 디렉토리에 내부에 pkl 확장자를 가진 파일이 존재한다. 이 파일은 학습이 진행된 뒤, 모델에 대한 모든 정보를 나타낸다. 해당 파일을 이용하여, test set을 대상으로 모델의 성능을 확인할 수 있다. Test 소스 코드 실행도 위와 동일하게 '!python3 + (파일 명)'을 입력하면 된다.

7. 참고문헌

[1] *Karen Simonyan, Andrew Zisserman* ,Very Deep Convolutional Networks for Large-Scale Image Recognition, eprint arXiv:1409.1556, 2014

[2] *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun*; Deep Residual Learning for Image Recognition The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

[3] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, Thomas Brox, FlowNet: Learning Optical Flow with Convolutional Networks, arXiv:1504.06852, 2015

[4] Yifan Hu, Yefeng Zheng ,A GLCM Embedded CNN Strategy for Computer-aided Diagnosis in Intracerebral Hemorrhage. arXiv:1906.02040. 2019

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going Deeper with Convolutions. arXiv:1409.4842. 2014