# 웹 기반 영상회의에서 딥 러닝 모델 적용을 위한 방법 연구

유상우, 고경찬, 홍원기
포항공과대학교
{rswoo, kkc90, jwkhong}@postech.ac.kr

# Research on Applying Deep Learning Methods on Web-based Video Conferencing Systems

Sangwoo Ryu[1], Kyungchan Ko[2], James Won-Ki Hong[2]
Graduate School of Artificial Intelligence, POSTECH[1]
Department of Computer Science and Engineering, POSTECH[2]

요 약

인공지능 기술의 발전으로 공장 자동화, 자율주행 등 다양한 산업 분야에서 인공지능이 사용되고 있다. 영상회의 시스템에서 또한 기존 알고리즘의 한계를 극복하기 위해 인공지능을 이용하는 기능들을 추가해 왔고, 대표적으로 초해상화(Super Resolution), 이미지 세분화를 이용한 가상배경 기능 등이 있다. 하지만 웹 기반 영상회의에서는 제한된 웹 환경으로 인해 이런 기능들의 적용이 제한된다. 본 논문에서는 이러한 웹 기반 서비스에서 딥 러닝 모델을 사용하는 기능을 제공하기 위해 웹 환경에서 딥 러닝을 적용하는 여러 방식을 소개하고, 각 방식에서 가상 배경 기능을 위해 사용하는 이미지 세분화 모델을 소개하고 성능을 평가한다. 마지막으로 웹 기반 영상회의에 딥 러닝 모델을 적용하기 위해 고려해야 하는 부분을 논의한다.

## Ⅰ. 서론

코로나-19 이후, 대면 업무를 지양하는 분위기가 형성되면서, Zoom, Webex 등 영상회의 서비스를 이용한 업무가 늘어나고 있고, 영상회의를 사용하는 고객의 요구사항 또한 늘어났다. 이에 따라 영상회의 서비스 제공자는 고객의 요구사항을 충족시키고, 사용 만족도를 높이기 위해 다양한 부가 기능을 추가하고 있다. 그 중 일부 기능들은 딥 러닝(Deep Learning, 심층학습)[1] 이전 방식보다 딥 러닝을 적용했을 때 더 좋은 성능을 보여주는데[24], 예를 들어 화질을 개선하기 위한 초해상화(Super Resolution), 얼굴의 정면을 자동으로 보여주는 얼굴 정렬(Face Alignment), 사용자의 배경을 바꾸는 가상 배경(Virtual Background) 등 다양한 기능에서 딥 러닝이 사용될 수 있다.

설치형 영상회의 서비스들은 Python/C++ 와 같은 프로그래밍 언어들을 직접적으로 사용할 수 있고 많은 라이브러리들이 존재해 딥 러닝을 쉽게 적용할 수 있다. 반면에 웹 앱들은 대부분 자바스크립트를 통해 기능이 제공되기 때문에, 웹 기반 영상회의 서비스들은 보다 제한적인 환경에서 모델 적용을 해야 한다. 이 논문에서는 딥 러닝의 여러 적용 사례들 중, 가상 배경 기능에 초점을 맞추어, 이 기능을 적용하기 위한 이미지 세분화(Image Segmentation) 작업을 수행하는 모델을 웹 환경에 적용하기 위한 방법들을 서버, 클라이언트라는 두 적용 위치에 따라 소개한다. 특히 딥 러닝 모델을 클라이언트에 적용하는 경우 WebRTC [13] 기반 영상회의 서비스인 Vmeeting[17]에 다양한 방법으로 적용 및 성능 평가를 수행한다. 마지막으로, 웹 기반 영상회의에 딥 러닝 모델을 적용하기 위해 고려해야 하는 부분을 정리하고 차후 성능 및 사용자 경험 개선을 위해 필요한 연구들을 제시한다.

## Ⅱ. 배경

### 1. 웹 환경에 딥 러닝 적용

웹 브라우저에 딥 러닝을 적용하는 방법은 모델을 적용하는 위치에 따라 나눌 수 있다.

딥 러닝 모델이 서버에 위치하는 경우에는, 브라우저와 별개로 서버의 자원(CPU, GPU)과 서버 측 코드를 이용해 딥 러닝 모델이 동작할 수 있기 때문에, 기존 방식을 그대로 이용할 수 있다. 다만 클라이언트 사이드와 입력 및 출력 데이터를 주고받아야 하기 때문에, 추가적인 대역폭 사용이 필요하다. 따라서 서비스 제공자의 입장에서 서버 구축 및 대역폭 확보를 위해 많은 비용이 소모될 수 있으며, 데이터의 종류에 따라 프라이버시 문제가 생길 수 있고, 서버에서의 연산으로 실시간 처리에 지연이 생길 수 있다.

딥 러닝 모델이 클라이언트에 위치하는 경우에는, 클라이언트의 자원을 이용해 딥 러닝 모델이 동작해야 하고, 영상회의 서비스 제공자가 그에 맞는 구현을 제공해야 한다. 클라이언트마다 서로 다른 기기를 사용하기 때문에, 모두에게 적절한 사용 환경을 제공하지 못할 수 있다는 단점이 존재한다.

#### 1) 자바스크립트 (JavaScript) 라이브러리

구글 오픈소스 프로젝트 TensorFlow[2]에서는 웹에서 머신러닝을 사용하기 위해 TensorFlow.js 라이브러리를 제공하고 있다. TensorFlow.js에서는 일반적인 사용 사례들인 이미지 분류, 객체 감지, 신체 분절화 등에 대해서는 선행 학습된 모델을 제공하고 있고, 이미 존재하는 모델을 재학습시킬 수 있도록 지원한다. TensorFlow.js에서는 저장소 및 수학 연산을 위해 여러

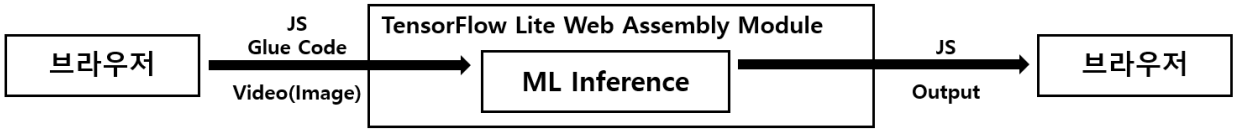


그림 1. TensorFlow Lite 웹 어셈블리 파이프라인

백엔드를 지원하는데 CPU, WebGL, WASM(Web Assembly) [18] 백엔드를 지원한다. 자바스크립트는 브라우저뿐만 아니라 React Native[19]를 통해 모바일 환경, Node.js[20]를 통해 IoT 기기로도 확장하여 사용할 수 있다.

TensorFlow.js 이외에도 Keras.js, ConvNetJS와 같은 자바스크립트 라이브러리에서 웹 브라우저에 딥 러닝을 적용하기 위한 도구들을 제공하고 있다[3].

### 2) 웹 어셈블리 (Web Assembly)

웹 어셈블리[4]는 웹에서 C/C++ 등으로 작성된 코드를 실행할 수 있도록 바이너리 형식으로 변환해, 가상 머신에서 실행하는 기술이다. TensorFlow에서 모바일 및 IoT 기기에서의 머신러닝 프레임워크인 TensorFlow Lite[2, 21]를 웹 어셈블리로 빌드하거나, 해당 라이브러리를 포함해 추론 과정 전체를 포함하는 프로그램을 웹 어셈블리로 빌드해 자바스크립트에서 해당 모듈을 불러와 모델을 사용한다. 모듈을 자바스크립트를 통해 불러오지만, 실제 연산은 순수 자바스크립트에서 수행하지 않기 때문에 빠른 속도를 보여준다. Google Meet에서는 실시간 영상에 기계학습 적용을 위한 구글 오픈소스 프레임워크인 Mediapipe를 웹 어셈블리로 빌드하여 가상 배경 기능을 위한 배경 분리를 딥 러닝 모델을 이용해 수행하였다[5]. 웹 어셈블리는 현재 Chrome, Safari, Firefox 등의 브라우저에서 2017년부터 지원하여 약 90%의 기기에서 지원을 하고 있다[6]. 자바스크립트를 통해 사용을 하기 때문에, 자바스크립트 라이브러리를 사용하는 경우와 마찬가지로 React Native 또는 Node를 이용해 모바일/IoT 기기에서 사용할 수 있다.

그림 1은 TensorFlow Lite를 웹 어셈블리로 빌드해 사용하는 경우의 파이프라인을 보여준다.

TensorFlow.js를 통해 WASM 백엔드를 사용하는 경우와 순수 웹 어셈블리를 사용하는 경우 모두 SIMD (Single Instruction, Multiple Data), 멀티 스레드(Multi Threads)와 같이 딥 러닝 연산을 가속화할 수 있는 추가적인 옵션을 선택할 수 있다는 장점이 있다.

## 2. 이미지 세분화 (Image segmentation)

이미지 세분화는 이미지에서 개체가 있는 위치, 모양, 픽셀을 판단해 픽셀 단위의 마스크를 출력하는 작업이다. 이미지 세분화는 자율 주행 자동차에서 도로, 차, 사람 등을 구분하기 위해 사용되거나, 의료 영상에서 특정 종양의 위치를 판단하는 등의 영역에서 사용될 수 있다. 영상 회의에서는 사람과 사람이 아닌 위치를 구분해 마스크를 만들고, 사람이 아닌 위치에 배경을 그리는 가상 배경 기능에 사용될 수 있으며, 그림 2는 실제로 영상회의에 가상 배경을 적용한 예시를 보여준다.

이미지 세분화를 위한 대표적인 딥 러닝 모델로는 DeepLab[7, 8]이 있으며, 특히 모바일 환경에 특화된 MobileNet[9, 10, 11] 또한 지속적으로 연구가 되고 있다.

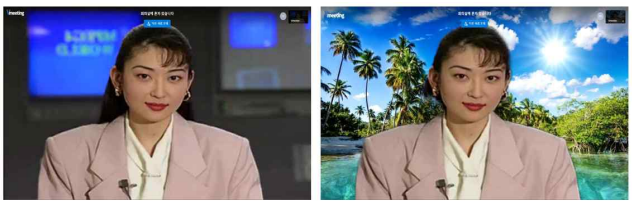

그림 2. 왼쪽 - 원본 영상, 오른쪽 - 가상 배경 적용

## 3. 웹 기반 영상회의 시스템

그림 3은 WebRTC 기반 영상회의 시스템의 기본적인 구조를 보여준다. 세션 기술 프로토콜(SDP, Session Description Protocol)[12]을 이용해 신호를 교환해 클라이언트들의 연결 지점을 알려주는 시그널링 서버와, 비디오, 오디오 등의 데이터를 전달하는 릴레이 서버로 이루어져 있다.

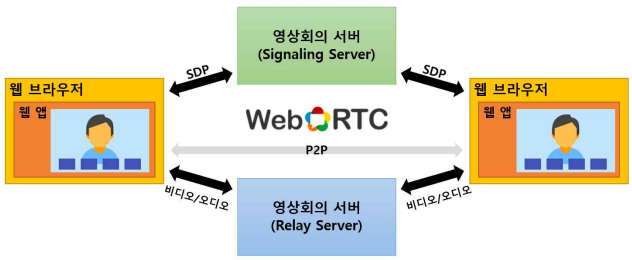


그림 3. WebRTC 기반 영상회의 구조

# Ⅲ. 성능 분석

이 장에서는 딥 러닝 모델을 서버에 적용했을 때 이론적 대역폭 사용을 보여주고, 딥 러닝 모델을 Vmeeting 클라이언트에 적용했을 때 여러 적용 방식 별 성능을 보여준다.

## 1. 서버 측 적용

딥 러닝 모델을 서버에 적용하는 경우, 클라이언트의 자원을 사용하지 않기 때문에 서버의 자원(CPU, GPU)의 성능에 따라 성능이 결정된다. 다만 데이터를 서버로 전송하고 클라이언트로 다시 전송을 해야 하기 때문에 초해상화(Super Resolution)[22, 23]와 같이 해상도와 직접적인 관계가 있는 작업들의 경우 서버에서 클라이언트 방향 대역폭 사용에 직접적인 영향을 주기 때문에, 이를 사용하기 위해서는 미리 대역폭 사용량에 관한 계산이 필요하다. 가상 배경 기능을 위한 배경 분리와 같은 작업들의 경우 해상도와 큰 관련이 없기 때문에, 대역폭에 영향을 주지 않는다.

### 2. 사용자 측 적용

딥 러닝 모델을 클라이언트에 적용하는 경우, 클라이언트가 사용하는 기기(데스크탑, 모바일 기기 등)의 자원에 따라 성능이 결정된다. 서버에 적용할 때와 마찬가지로 수행하려는 작업과 방식에 따라 대역폭 사용에 영향을 준다. 초해상화(Super Resolution) 작업의 경우, 영상회의에서 처음 비디오를 만들어내는 과정에서 초해상화를 수행할 때는 클라이언트에서 서버로, 서버에서 클라이언트로 대역폭 두 곳 다 영향을 주지만, 영상회의에서 영상을 받는 쪽에서 화면에 표시되는 사람의 영상에 대해서만 수행할 때는 일반 영상회의와 큰 차이가 없다.

그림 4는 N명이 참여하고 참여자에서 영상회의 서버로 전송하는 기본 대역폭이 M kbps인 영상회의에서 영상의 너비, 높이를 2배씩 증가시키는 2x Super Resolution을 수행할 때, 기본 상태와 서버에 적용하는 경우, 클라이언트는 보내는 시점, 받는 시점에 적용하는 경우 총 4가지 경우에서 예상 대역폭 사용을 보여준다.

아래에서는 가상 배경 기능을 위해 각 적용 방식에서 제공하거나 사용할 수 있는 이미지 세분화(Image Segmentation) 모델을 WebRTC[13] 기반 영상회의인 Vmeeting에 적용했을 때의 성능을 보여준다. 이 모델은 영상의 해상도를 변경하지 않기 때문에 대역폭에 큰 영향을 주지 않는다.

FPS 및 추론 시간은 2.90GHz i5-9400F CPU, 16.0GB RAM 데스크탑 PC의 Chrome 브라우저에서 100초 간 수행한 결과의 평균을 이용하였다. 라이브 스트리밍 지원 프로그램인 OBS Studio[14]에서 제공하는 가상 카메라를 이용하였으며, JVT(Joint Video Team) 테스트 영상들 중 영상회의와 유사하게 실내 환경에서 상체만 나오는 10초 길이의 “Akiyo” 영상을 반복 재생하였다.

1) 자바스크립트 - TensorFlow.js

TensorFlow.js에서는 사람-배경 분리를 위한 모델로, Body-pix[15]를 제공하고 있으며, 딥 러닝 모델로 ResNet-50[16]과 MobileNet-V1[9]을 선택할 수 있다. 각 모델에 대해 출력 스트라이드 (Output Stride), 양자화 (Quantization) 정도를 조절할 수 있으며, MobileNet-V1의 경우 컨볼루션 (Convolution) 연산에서의 채널 수를 조절해 전체 모델의 크기를 0.6MB부터 13MB까지 조정할 수 있다. 제공하는 모델 이외에도 필요한 경우 커스텀 모델을 불러와 사용할 수 있다.

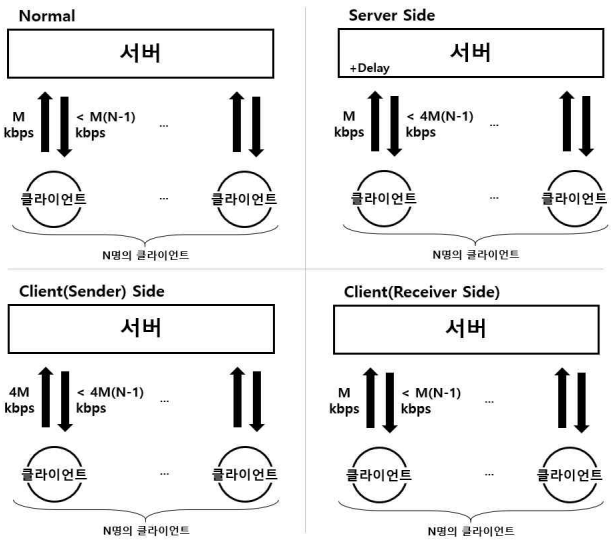


그림 4. 2x Super-Resolution 연산에서 예상 대역폭 사용

TensorFlow 공식 문서에서는 중간 크기 모델 (~100-500M multiply-adds)은 WASM 백엔드가 WebGL 백엔드보다 느리고, 가벼운 모델 (~20-60M)은 WASM 백엔드가 빠르다고 설명한다.

표1은 중간 크기 모델인 MobileNet-V1을 적용하고 WebGL 백엔드를 사용한 경우 영상회의에서 배경 분리 작업의 성능을 보여준다.

| Multiplier | quantBytes=2 | | | quantBytes=1 | | |
|---|---|---|---|---|---|---|
| | 크기 | FPS | 추론 시간 (ms) | 크기 | FPS | 추론 시간 (ms) |
| 0.75 | ~2MB | 8.06 | 80.28 | ~1MB | 7.97 | 76.94 |
| 0.5 | ~2MB | 8.93 | 70.32 | ~0.6MB | 8.94 | 67.93 |

표 1 . TensorFlow.js MobileNet-V1 WebGL 백엔드 성능

모델의 채널 수(Multiplier), 양자화(quantBytes) 정도를 조정해 모델의 크기와 연산의 수를 조정하여 추론 속도(추론 시간)를 조절할 수 있는 것을 확인할 수 있다. 다만 이에 따라 추론 정확도가 달라질 수 있다.

2) 웹 어셈블리 (Web Assembly)

웹 어셈블리를 이용하는 경우 사용하려는 머신러닝 라이브러리에 맞는 커스텀 모델을 사용해야 하는데, TensorFlow Lite를 사용하는 경우, TensorFlow Lite에서 제공하는 이미지 세분화 모델 등 TensorFlow Lite에서 동작할 수 있는 모델들을 사용할 수 있다.

표 2는 Google Meet에서 Apache 2.0 라이센스로 제공했던 MoblieNetV3-small 개선 모델을 적용한 영상회의에서 배경 분리 작업의 성능을 보여준다.

| Non-SIMD | | | SIMD | | |
|---|---|---|---|---|---|
| 크기 | FPS | 추론 시간 (ms) | 크기 | FPS | 추론 시간 (ms) |
| 400KB | 16.97 | 8.38 | 400KB | 17.45 | 6.91 |

표 2 . 웹 어셈블리 - Google Meet Segmentation 모델 성능

표 3은 TensorFlow Lite에서 이미지 세분화를 위해 제공하는 DeepLab-V3[7]을 적용한 영상회의에서 배경 분리 작업의 성능을 보여준다.

| Non-SIMD | | | SIMD | | |
|---|---|---|---|---|---|
| 크기 | FPS | 추론 시간 (ms) | 크기 | FPS | 추론 시간 (ms) |
| 2.7MB | 3.16 | 260.44 | 2.7MB | 8.11 | 63.98 |

표 3 . 웹 어셈블리 - DeepLab-V3 모델 성능

Google Meet Segmentation 모델은 양자화 및 최적화가 수행된 모델이기 때문에 상대적으로 크기가 큰 DeepLab-V3 모델보다 더 좋은 성능을 보여주었고, 두 모델 모두 SIMD를 사용했을 때 더 좋은 성능을 보여준다.

## Ⅳ. 결론 및 향후 연구

본 논문에서는 기존 딥 러닝을 사용하는 일반적인 환경과 달리 제한적인 웹 환경에서의 딥 러닝 적용을 위해, 웹 기반 영상회의에서의 딥 러닝 모델 적용 방식들을 서버와 사용자라는 두 적용 위치에 따라 설명하였다. 특

히 사용자 측 적용을 했을 때 사용할 수 있는 자바스크립트 라이브러리와 웹 어셈블리 두 방식을 소개하였고, 각 방식을 사용했을 때 대역폭 사용 및 성능을 분석하였다.

웹 기반 영상회의에 딥 러닝을 적용하기 위해서는 다양한 요소들이 고려되어야 한다. 먼저 딥 러닝 모델이 수행하려고 하는 작업에 따라 자원 사용 및 네트워크 대역폭 사용량이 달라질 수 있기 때문에, 이를 미리 분석하고 적절한 모델 적용 위치를 선택하는 것이 필요하다. 또한 본 논문에서 소개한 TensorFlow.js의 백엔드들과 웹 어셈블리를 이용한 머신 러닝 추론 방식들은 모델 크기 및 계산 량에 따라 속도가 달라지기 때문에, 사용하려는 모델에 맞게 모델 적용 방식을 선택해야한다. 딥 러닝 모델 연구가 성능을 높이기 위해서만이 아닌, 모델을 최적화하기 위한 연구들도 함께 진행이 되고 있다. 영상회의와 같은 웹 서비스에서는 실시간으로 이미지 처리가 수행되어야 하기 때문에, 성능과 최적화 두 부분을 모두 고려한 딥 러닝 모델 선택이 필요하다. 마지막으로, 딥 러닝 모델의 결과물을 정확하고 자연스럽게 보여주기 위한 방법을 연구할 필요가 있다.

클라이언트에서 딥 러닝 모델을 사용하는 경우에는 모델의 성능이 클라이언트 기기 성능에 의존하기 때문에, 실시간 영상회의 환경에서 사용자 경험을 해치지 않기 위해 환경에 적응할 수 있는 딥 러닝 모델 적용 연구가 필요하다. 이와 동시에 MobileNet 모델과 같이 저성능 디바이스용 모델들도 지속적인 개선 연구가 필요하다.

## ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발, 2017-0-01633, 대학ICT연구센터육성지원사업).

## 참 고 문 헌

[1] LeCun, Y., Bengio, Y. & Hinton, G. "Deep learning," Nature 521, pp. 436 - 444, 2015, (https://doi.org/10.1038/nature14539).

[2] Martín Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, (https://tensorflow.org).

[3] Ma, Y. et al.. "Moving Deep Learning into Web Browser: How Far Can We Go?," World Wide Web Conference, pp. 1234 - 1244, Association for Computing Machinery, 2015

[4] Haas, A., et al. "Bringing the Web up to Speed with WebAssembly," SIGPLAN Not., 52(6), pp. 185 - 200, 2017

[5] Background features in google meet, powered by web ml. (2020, October 30). Retrieved March 23, 2021, from https://ai.googleblog.com/2020/10/background-features-in-google-meet.html

[6] Can I use... support tables for HTML5, css3, etc. (n.d.). Retrieved March 23, 2021, from https://caniuse.com/wasm

[7] Chen, L.C. et al. "Rethinking Atrous Convolution for Semantic Image Segmentation," arXiv e-prints, arXiv:1706.05587, 2017.

[8] Chen, L.C. et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," arXiv e-prints, arXiv:1802.02611, 2018.

[9] Howard, A. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv e-prints, arXiv:1704.04861, 2017.

[10] Sandler, M. et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv e-prints, arXiv:1801.04381, 2018.

[11] Howard, A. et al. "Searching for MobileNetV3," arXiv e-prints, arXiv:1905.02244, 2019.

[12] Handley, Mark et al. "SDP: Session Description Protocol," IETF, doi:10.17487/RFC4566, RFC 4566, July 2006.

[13] A. Bergkvist et al. "WebRTC 1.0: Real-time communication between browsers," W3C Working Draft, Feb. 2015.

[14] OBS studio, https://obsproject.com/

[15] Tensorflow. BodyPix - Person Segmentation in the Browser, https://github.com/tensorflow/tfjs-models/tree/master/body-pix

[16] He, Kaiming et al. "Deep Residual Learning for Image Recognition," pp. 770-778. 10.1109/CVPR.2016.90, 2016.

[17] Vmeeting, https://vmeeting.io

[18] Haas, A. et al. "Bringing the Web up to Speed with WebAssembly," SIGPLAN Not., 52(6), pp. 185 - 200, 2017.

[19] React Native, https://reactnative.dev/

[20] Node.js, https://nodejs.org/

[21] TensorFlow Lite, https://www.tensorflow.org/lite

[22] Yang, W. et al.. "Deep Learning for Single Image Super-Resolution: A Brief Review," arXiv e-prints, arXiv:1808.03344, 2018.

[23] Anwar, S. et al. "A Deep Journey into Super-resolution: A survey," arXiv e-prints, arXiv:1904.07523, 2019.

[24] Alom, Md. Zahangir et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures," Electronics. 8. 292. 10.3390/electronics8030292, 2019.