Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-4, April 2026
ISSN: 3049-2602

AI-DRIVEN DEEPFAKE DETECTION SYSTEM

M. V. Gnaneshwar[1], Mrs. K. Neeharika[2]
[1]Student, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India
[2]Assistant Professor, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India
Email id: munagalagnaneshwar@gmail.com, karra.neeharika@aliet.ac.in

Abstract: The widespread use of AI-produced synthetic material, otherwise known as deepfakes, is an increasing challenge to digital trust, facts, and personal confidentiality. The project includes a design and a realization of a web-based deepfake detection network using a three-model ensemble architecture to provide relevant classification of images and videos as real or fake. The three complementary deep learning models that are the core of the detection pipeline are a custom-trained convolutional neural network that is fine-tuned to binary classification of real and AI-generated content, with a validation of 97.61 per cent, and two pre-trained models that specialize in the task of output recognition in modern generative AI systems each making an equal contribution to a confidence-weighted ensemble decision. To analyse images, uploaded media is evaluated concurrently using all three models, and results in a final classification of AI Generated, Real, Uncertain, Likely Deepfake, or Likely Real, with each model presented with a confidence score. In the case of video contents, frame-level extraction is undertaken through the application of computer vision algorithms whereby each sampled frame undergoes assessment independently and the findings are pooled together into a semblance score. The heatmap based visual explainability is also integrated in the system and identification of the spatial areas that have the greatest effect to the model prediction were highlighted, and this increases interpretability and user trust. The backend is run on a cloud-based GPU infrastructure and it is exposed as a RESTful API and can also be used to analyze YouTube video URLs, process images in batches, and cumulative detection analytics. The front end gives the user a responsive, easy to use interface where they can upload and analyze media on-the-fly. This framework proves that ensemble-based deep learning systems, which integrate task-specific and general-purpose models, can achieve a high level of detecting deepfakes, which provides a convenient input in the fight against the misuse of generative AI technologies.

Keywords: Deepfake Detection, Ensemble Learning, Convolutional Neural Networks (CNN), Computer Vision, Explainability, Heatmap Visualization, Image Classification, Video Analysis, Frame-Level Processing, Generative AI, Media Authenticity, Artificial Intelligence, Cloud Deployment, RESTful API, Cybersecurity.

1. INTRODUCTION

The development of easily available generative AI applications has led to the easy realization of producing photorealistic synthetic images and videos by people. A technology based on the deep neural network synthesizing or manipulating faces, voices, and scenes called Deepfake has gone beyond research novelty and become a mainstream issue. Cases of deepfake-based fraud, political fake news, and un-consented intimate pictures have increased the need to have effective and easily accessible detection mechanisms.

Conventional methods of media forgery detection were based on manual inspection and signal processing of a signal using rules. They do not work on the latest generative adversarial networks (GANs) and diffusion models that generate imperceptible artifacts almost cannot be seen with the human eye. This is an incentive to work towards deep learning systems that are automated, that can differentiate authentic content and synthetic content on large scales.

The given project introduces a deepfake-based detection platform on an end-to-end basis. The system accepts individual images, video files, batches of images and URLs of YouTube videos as the input, and gives a classification judgment (AI GENERATED, LIKELY Deeefake, REAL IMAGE, LIKELY Real, or UNCERTAIN) with confidence scores and per-model probability distributions and Grad-CAM attention heat maps as outputs. A long-term analytics display provides the data of historical scans into a summary and shows the usage trends with time.

The Python FastAPI application is the backend REST API, which consists of EfficientNet and ResNet50 backend classifier ensembles to detect it. The frontend is a single-page modular application in vanilla JavaScript and with Chart.js to do interactive visualizations. Ngrok is used to deploy the system but exposes the local backend to a publicly accessible HTTPS endpoint, thus it can be immediately used without cloud infrastructure.

2. Literature Survey

DeepFake detection studies have gained additional ground with the improvements in the generative modeling. The initial research to detect GAN-specific frequency artifacts in the fourier domain. Later experiments were able to show that CNNs that had been trained with face-centric data sets were able to infer a wide category of synthetic content. Classifiers trained on the EfficientNet framework have demonstrated a significant level of performance on image level deepfakes benchmarks because they scale depth, width, and resolution compositely.
In the case of video material, temporal inconsistency analysis has been found to be effective. Frame-level CNN classifiers and re-combined score over aggregated confidence on estimated frames are able to identify manipulated sequences even in cases where frames are uncertain. It has adopted Grad-CAM (Gradient-weighted Class Activation Mapping) to understand the choices made by CNN, and indicate the areas of the image that have the most impact on a classification decision, which otherwise is a black-box process.
The known literature has found several common limitations: systems tend to handle only one modality (image or video), cannot be explained, and need special cloud infrastructure with authentication overhead. There is limited information on platforms providing a combination of detection and interpretability delivered in one, accessible interface. In the literature surveyed, the following are the important themes:

- Evolution of GAN and Diffusion-based Deepfake Generation Techniques.
- CNN and EfficientNet-based Image Authenticity Classification.
- Timed Frame Analysis of Video Deepfake Detection.
- Grad-CAM and explainable AI mechanisms of model interpretability.
- Ensemble Learning Strategies of better Detection Robustness.
- SaaS Architecture Frameworks in AI-Driven Media Analysis Solutions.

The mentioned gaps are filled in the current work by incorporating multi-modal detection (image, video, batch, YouTube), Grad-CAM visualization, ensemble inference, and an analytics layer into a single deployable model with a small lightweight dependency-free frontend.

3. Proposed System

The proposed AI-Driven Deepfake Detection System is a client-server web application structured around a Python backend and a vanilla JavaScript frontend. The system is designed for multi-modal media analysis and provides a unified interface for four distinct detection workflows.

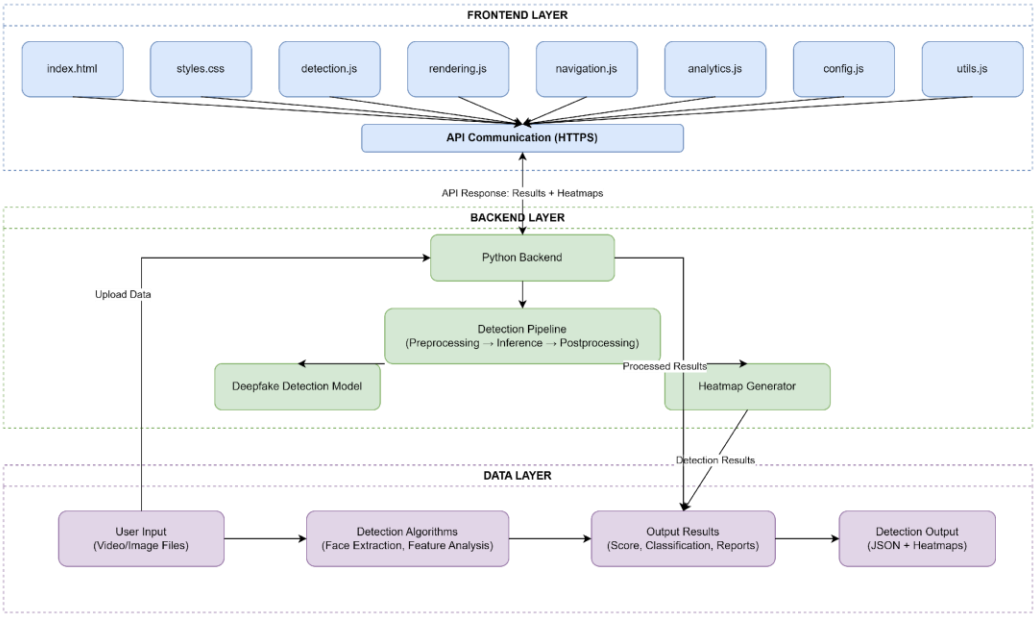

Fig 1: Proposed System

The backend opens a RESTful API through FastAPI that has endpoints (single image detection, /detect), (video), /detect_video, endpoints (batch image processing, /detect_batch), (mail), /analyze_youtube, endpoints (Grad-CAM heatmap generation, /heatmap and /heatmap video, endpoints (analytics), and a health endpoint (/health). Every endpoint receives requests with a custom ngrok-skip-browser-warning header, which will reduce redeploying on a tunneled basis to aspects of warning about browser-level redirects.

The frontend has six JavaScript modules, namely, config.js (backend base URL and easy state), navigation.js (page routing, drag-and-drop, file preview), detection.js (coordination of API calls), rendering.js (assembling the result card and Integrating Chart.js), analytics.js (populating the dashboard and building the history table), and utils.js (shared utility, such as verdict classification, toast not This modular break has made it possible to maintain and test every concern separately.

4. Methodology

The workflow of the system is put in place in a streamlined and effective manner beginning with the user input and proceeds to processing, detection, and presentation of results. Frontend and backend are asynchronous, and generation of heat maps works in parallel to save time of waiting and enhance user experience.

1. File Submission

   The user is able to drag and drop an image or video into the interface or direct a YouTube link. The frontend does the validation of the input and ready the data to be passed to the backend. It relies on Form Data in order to upload files and the use of JSON requests in case of YouTube links.

2. Detection Request

The frontend then provides the input to the backend by using certain API endpoints depending on the type of input (image, video, batch, or YouTube). The request is sent to the backend which in turn processes the request and gives the final verdict, confidence score, AI probability, real probability and optional per-model results.

3. Heatmap Generation
Once the detector reaches a detection result, the system will make a parallel request to create a heatmap. It is this heatmap that shows the areas that had an impact on the decision made by the model. It is produced through Grad-CAM on a ResNet50 model and presented in the form of an overlay and raw image. In case of a failure in generation of the heatmaps, the key detection result is shown immediately.

4. Model Processing (Backend)
The models used in the backend are EfficientNet and ResNet50. All the models indicate the real or AI generated content. This is a combination of these predictions by weighted method to give an eventual probability score. Using this score the content falls into the AI Generated, Likely Deepfake, Uncertain, Likely Real or Real Image.

5. Video Analysis
With the case of video inputs, one frame is extracted at intervals and processed separately. These are then summed up in order to establish overall authenticity. The system also computes the percentage of frames that are probably AI generated and generates a sequence of AI probability on the video.

6. Displaying Results
The frontend renders the results into a user termic format which consists of a clear verdict, confidence indicators, probability charts and heatmap visualization. In the case of videos, a time graph is used to display the changes in AI probability as a matter of time.

7. History and Analytics
Every scan is stored locally in a browser containing information on the file name, type of media, verdict and score confidence and date. The interface allows the user to see previous analyses. A section on analytics gives information on usage and recent detections, and where there are backend analytics provided, they contain more precise and aggregated data.

5. Proposed System Software Implementation & Results

Frontend Implementation

The frontend will be a single-page multi-page application (SPA), which takes four pages to navigate the page: Image Detection, Video Detection, Batch Detection and a YouTube Analysis, and an Analytics dashboard. Navigation navigation.js is also in charge of navigation which is controlled by showPage() which switches CSS classes on page elements and tab buttons and has no JavaScript framework and build step. To avoid the creation of memory leaks as a result of the use of the canvas, chart rendering with a shared charts4 registry defeats that chart instances are created and destroyed with each result update. This whole procedure is taken care of within the helper function mkChart in the file utils.js that takes care of how the chart is made, updated and shown. GradCam heatmap card is returned to the heatmapCard() method of rendering.js which inserts the two img elements (overlay and raw heatmap) and a toggle row and manages its appearance by turning its elements on and off by id. Analytics are produced using analytics.js to create the analytics dashboard. It takes as an input either a data object as returned by the backend /analytics endpoint or a calculation by the fallback of local storage history, and uses this to fill out KPI counter

elements, progress bars representing media type distribution, a doughnut chart representing their verdict distribution, a weekly scan bar chart with that data watering wheeled in the scan count, and a recent detections table with per-row confidence bar cells and badges on verdict.

Backend Implementation

The backend is a FastAPI application that is served on a Python runtime, and ngrok tunnels with the use of HTTPS. BACKEND variable in the config.js encodes the currently active ngrok subdomain URL and it has a single configuration point of changing the environment. ngrok-skip-browser-warning header is associated with HDR constant because it is necessary with all the fetch calls to avoid the interception of redirects. The current status and the device string (cpu or cuda) is then returned by the health endpoint and it is shown in the status indicators by the frontend. The detect endpoint takes the multipart/form-data POST that contains a file field and processes the image, then it applies ensemble inference and generates a payload in the shape of a JSON. The detect video endpoint also takes a video field, demystifies the field to frames and performs per-frame inference and gives out aggregated results with a timeline array. The heatmap endpoint has resorted to the uploaded image being passed through ResNet50 using Grad-CAM hooks to create a colormap overlay. The output of overlaying the b64 of heatmap on the base image and the result of the b64 of the heatmap is returned in form of base64. The heatmap video endpoint takes N best frames according to the AI probability in the video and works with heatmaps of the frames. It makes these available as [array] where the index of frames is accompanied by the metadata of the metadata that indicates the AI probability.

Findings and Case Study Results

In the set of evaluations, AI-generated images of Stable Diffusion and DALL-E, real photographs, deepfakes video clip samples on publicly available datasets, and video recording were evaluated. The principle testing findings can be summarized as below:

• The reports of modern AI diffusion models were categorized as AI GENERATED, AI probability was always above 0.80 when there was an ensemble of images. Real images were also divided into categories of REAL IMAGE having real probability larger than 0.78 (but border cases low processed photo images, artistic filters) as well as LIKELY REAL and UNCERTAIN. Deepfake video clips also recorded per-frame results in the domain of a range between 0.65 and 0.95.

• Grad-CAM Properly highlighting facial locations, eye areas, and hairline boundaries were among the central areas of attention to AI-generated images, which are also known to cause GAN artifacts.

• Both Batch processing of 20 image sets, which completed within respectable latency and the results of each image have been returned as well as aggregate summary statistics in one response. YouTube analysis was effective in extracting metadata of videos, sampling frames, and providing a verdict with timeline information of videos that were publically available.

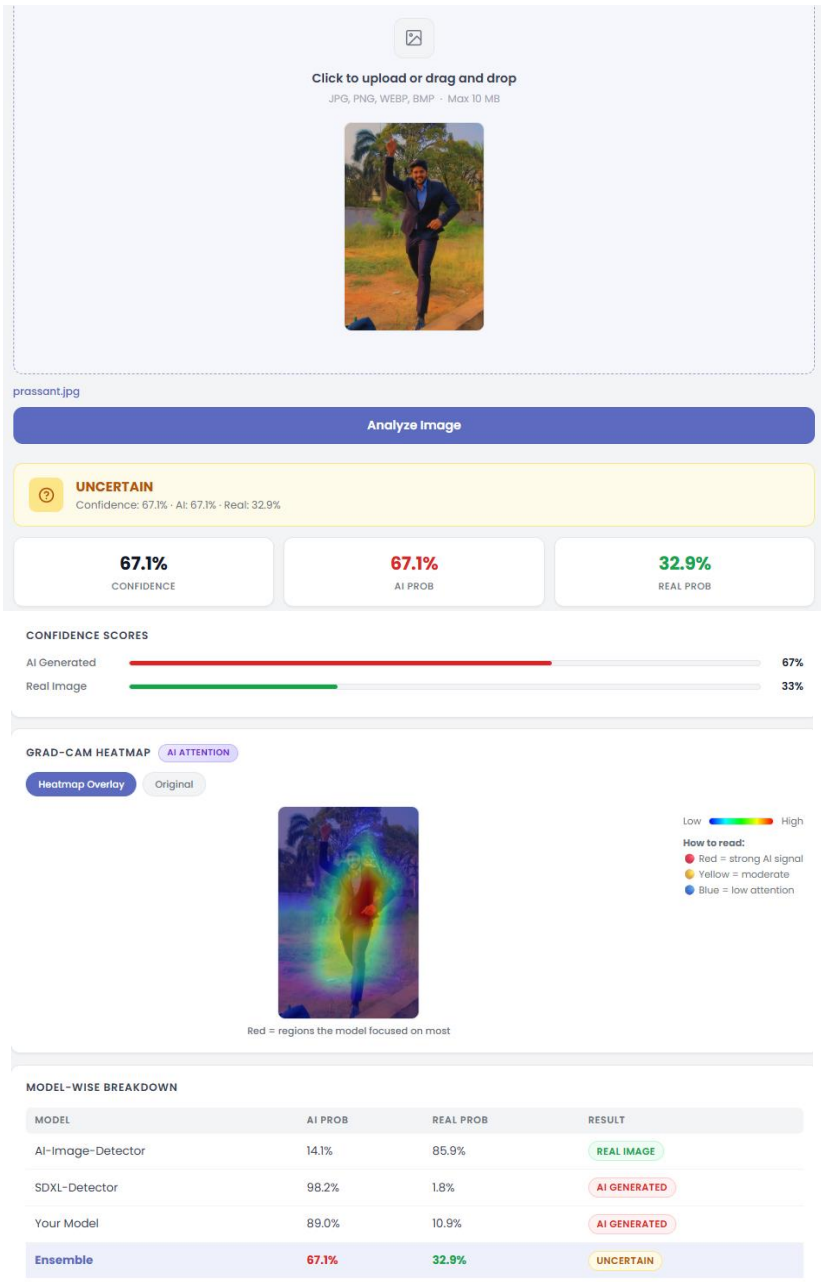

Fig 2. Test Case on Image.

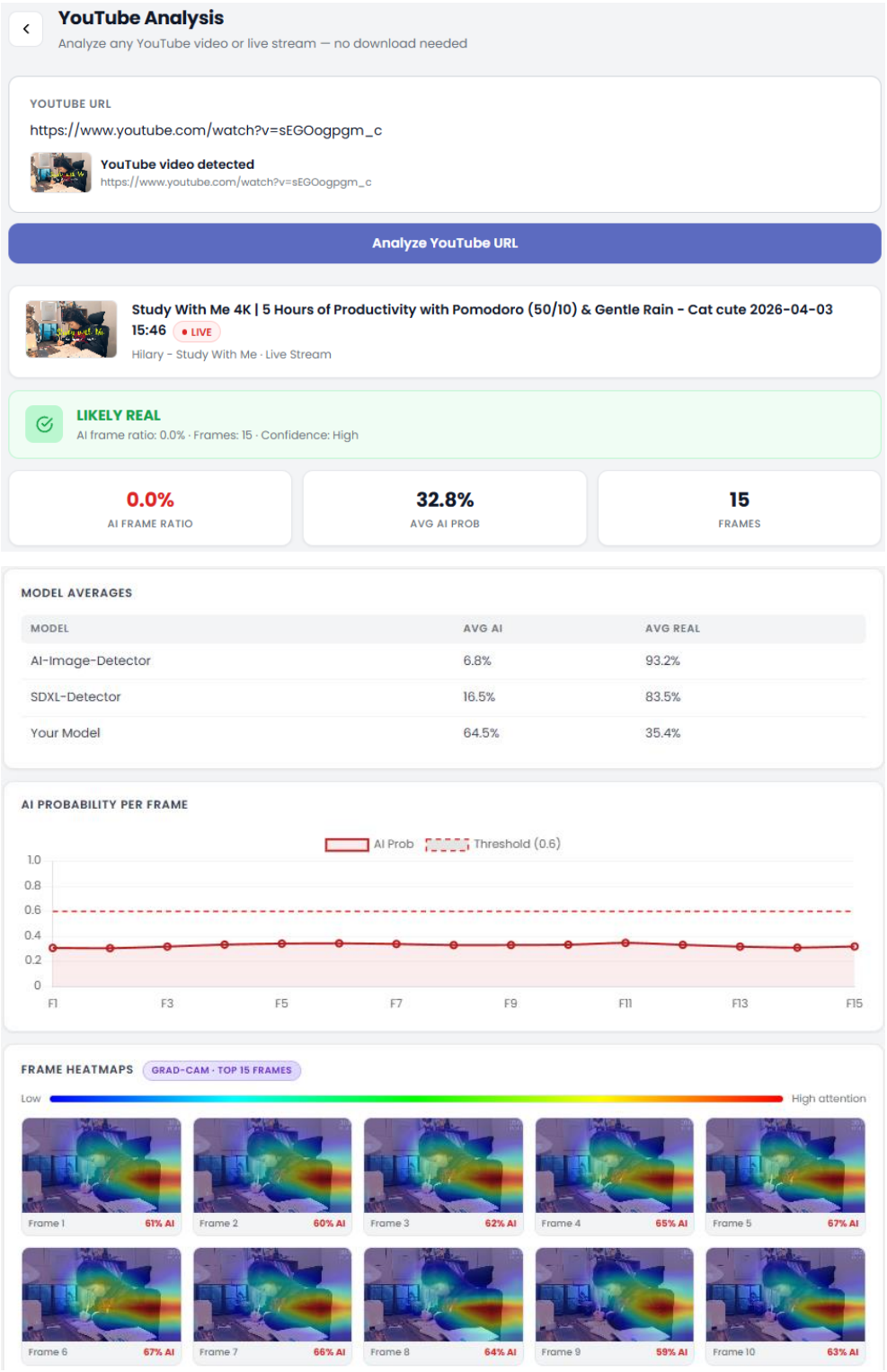

Fig 3: Test Case on live YouTube Video

6.CONCLUSION

The present paper introduced AI-powered Deepfake Detection System that is able to conduct image, video, and batch uploading, and YouTube streams processing based on the ensemble of CNN-based classifiers with Grad-CAM explainability. The system meets its design requirements: it was designed as multi-modes to achieve precise detection, the model has been interpreted transparently through heatmap visualization, persistent analytics, and the system can be easily deployed without any cloud subscription or authentication servers.

The interactive, high-quality computer-assisted-by-artificial-intelligence JavaScript frontend has shown that a high-end, interactive free-of-bloodthirsty-frameworks web application can be engineered without complicated deployment, and that will be even easier to maintain. The FastAPI server offers sufficient isolation of the inferences, heatmap imaging, and analytics issues, so computationally demanding elements can scale separately.

The direction of work in the future will be on three directions. To begin with, the model ensemble can be extended to the use of transformer-based models (Vision Transformers (ViT) and CLIP-based classifiers) to enhance resistance to content generated by diffusion models. Second, an option to use a real-time analysis of webcams streams to provide a deepfake in video calls. Third, deploying the backend to autoscaling persistent cloud (AWS Lambda or Google Cloud Run) infrastructure and allowing many users to share it without having to manage ngrok on its own. The capability to integrate with browser extensions and with social media platform APIs further is also an interesting avenue of expansion into the real world.

7.REFERENCES

1. Ian J. Goodfellow et al., "Generative Adversarial Networks," Advances in Neural Information Processing Systems (NeurIPS), 2014.
2. M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML, 2019.
3. R. R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," ICCV, 2017.
4. Rossler et al., "FaceForensics++: Learning to Detect Manipulated Facial Images," ICCV, 2019.
5. S. Agarwal et al., "Protecting World Leaders Against Deep Fakes," CVPR Workshops, 2019.
6. FastAPI Documentation, Sebastián Ramírez, Available: https://fastapi.tiangolo.com, Accessed 2025.
7. OpenCV Documentation, "OpenCV for Image and Video Processing," Available: https://opencv.org, Accessed 2025.
8. Chart.js Documentation, Available: https://www.chartjs.org, Accessed 2025.
9. yt-dlp, "YouTube Download Library," Available: https://github.com/yt-dlp/yt-dlp, Accessed 2025.
10. K. He et al., "Deep Residual Learning for Image Recognition," CVPR, 2016.