We demonstrate Rapsai, a visual programming platform that aims to streamline the rapid and iterative development of end-to-end machine learning (ML)-based multimedia applications. Rapsai features a node-graph editor that enables interactive characterization and visualization of ML model performance, which facilitates the understanding of how the model behaves in different scenarios. Moreover,
the platform streamlines end-to-end prototyping by providing inter-
teractive data augmentation and model comparison capabilities
within a no-coding environment. Our demonstration showcases
the versatility of Rapsai through several use cases, including vir-
tual background, visual effects with depth estimation, and audio
denoising. The implementation of Rapsai is intended to support ML
practitioners in streamlining their workflow, making data-driven
decisions, and comprehensively evaluating model behavior with
real-world input.

CCS CONCEPTS
• Computing methodologies → Visual analytics; Machine learn-
ing; • Software and its engineering → Visual languages.

KEYWORDS
visual programming, node-graph editor, deep neural networks, data
augmentation, deep learning, model comparison, visual analytics

ACM Reference Format:

1 INTRODUCTION
Despite the availability of user-friendly tools that have facilitated
the development of machine learning (ML) models in language [2,
12–14] and image classification [1, 5, 8, 10] domains, there remains a
significant gap in the availability of tools that support real-time mul-
timedia applications. Current tools fall short in providing efficient
means for handling visual and audio data from real-world sources,
such as camera streams, and enabling interactive experimentation
with data augmentation and model comparison. Moreover, there is
a lack of tools to facilitate the prototyping with multiple ML models
and effective communication for model performance and feedback
in the computer vision and audio research.

In this demonstration paper, we present Rapsai [3], a visual
programming platform that streamlines the iterative development
of perception pipelines through a node-graph editor. It enables
ML practitioners to interactively gain insights into model behavior
and assess trade-offs through data augmentation modules. Rapsai
accelerates the systematic comparison of different deep learning
models and enables users to explore a variety of configurations in
end-to-end pipelines.

2 SYSTEM OVERVIEW
Rapsai is a cross-platform application that operates within a web
browser, providing users with a convenient and accessible platform.
Fig. 1 presents an overview of the Rapsai interface, which consists
of four coordinated panels: (a) Nodes Library, (b) Node-graph Editor,
(c) Preview Panel, and (d) Node Inspector. Fig. 2 illustrates a few
example elements of each panel and we encourage readers to refer
to our full paper [3] for more details.

Real-time performance is a key objective of the Rapsai framework
to enable the efficient comparison of deep learning models and
adapt to real-time multimedia applications. To achieve this goal, we
employ GPU computing throughout the entire pipeline, including
model inference, data augmentation, and WebGL rendering, using
TensorFlow.js [6, 9]. Rapsai uses an off-screen WebGL context for
model inference and automatically evaluate the uploaded Keras or
Graphdef model when the input changes or the pipeline is updated
in the Node-graph Editor. In the data augmentation nodes, which
involve frequent image processing, we use hardware-accelerated
HTML canvas or TensorFlow.js image operations on the GPU to

Figure 2: Core views of Rapsai: (a) Node Library contains five
categories of nodes and a search bar to filter desired nodes.
(b) Node-graph Editor allows users to build and adjust a mul-
timedia pipeline by dragging and dropping nodes from the
nodes library, or suggesting available nodes when dragging
an edge from an existing node. (c) Preview Panel depicts vi-
sualization of every visible node in the Node-graph Editor.
(d) Node Inspector shows advanced properties of a selected
node, e.g., users can upload their own images for rapid test-
ing when selecting an input image node.
process images or videos in real-time, as opposed to relying on CPU-hosted arrays. Additionally, in shader processing nodes, we create WebGL canvases to visualize the results utilizing fragment shaders. It is worth noting that audio mixing and volume adjustments are currently performed on the CPU.

3 USER JOURNEY OF RAPSAI

To build a machine learning application in Rapsai, users may opt to either create a new pipeline from scratch, or copy an existing pipeline to start with from a list of public pipelines. We use the screenshots of the node-graph editor and preview panels in Fig. 2 as an example. Initially, the user drags an input camera node to the node-graph editor; then the user attaches an edge from the output dot of the input camera node, and selects a new node of image processing. This node allows for real-time adjustments of the input camera stream’s cropping and brightness. Next, the user creates a body segmentation node from the node library and connects it to the output node. The user then adds another input image node and uploads the background image with the node inspector. Finally, the user creates an image mixer node and blends the segmented image with the background image to generate new real-time visual effects.

4 RAPSAI USE CASES

With Rapsai, machine learning practitioners are able to efficiently author a diverse array of multimedia pipelines within a short timeframe of less than 15 minutes. This includes tasks such as real-time body segmentation, image classification, super-resolution, and depth estimation, as detailed in [3]. In this demonstration, we present four exemplary use cases with Rapsai.

4.1 Portrait Depth with Relighting Effect

The portrait depth pipeline uses two publicly-available models from TensorFlow Hub: AR portrait depth¹ and MediaPipe segmentation². The depth estimation model uses a single color portrait image as input and generates a depth map, which estimates the distance of each pixel to the camera.

4.2 Scene Depth for Real-time Visual Effects

The scene depth pipeline uses two scene depth estimation models based on ResNet and U-Net. Like the portrait depth model, these models predict a depth map from a generic color image and can be further applied to a variety of augmented reality applications such as occlusion-aware rendering, rain effects, and fog effects [4]. In Fig. 4, we present a pipeline that generates real-time depth-aware fog effect from an input RGB image.

4.3 Matting for Virtual Conferences

In the context of virtual background in remote video meetings, we present a matting pipeline that compares the performance of two alpha matte models. By leveraging the publicly available Mediapipe segmentation model along with two alpha matting models [7], this pipeline enhances the accuracy of segmentation in video conferencing scenarios and downstream applications [11]. The matting models take as input the original image and a rough segmentation mask from a body segmentation model, and produce a refined segmentation mask as output. To effectively evaluate the benefits of each model, Rapsai allows ML practitioners to compose the refined segmentation masks with virtual backgrounds in Fig. 5.

4.4 Audio Denoising for Remote Communication

In the audio denoising pipeline, we provide users with the ability to record their own voice and qualitatively compare the performance of two audio denoising models. Additionally, users can mix their recordings with background noise data, change volumes of the input, and provide qualitative feedback for the models directly in Rapsai using a Survey node, which is powered by Google Forms.

---

¹Portrait Depth API: https://tfhub.dev/tensorflow/tfjs-model/ar_portrait_depth/1
²MediaPipe API: https://tfhub.dev/mediapipe/tfjs-model/selfie_segmentation/general

Figure 3: Comparison between two portrait depth models with depth visualization and relighting shader effects.

Figure 4: Comparison between two scene depth models using both depth visualization and visual effects.
We would like to extend our thanks to all authors of the main Rapsai paper, including Scott Miles, Maria Kleiner, Yinda Zhang, Anuva Kulkarni, Xingyu “Bruce” Liu, Ahmed Sabie, Sergio Escolano, and Abhishek Kar, as well as contributors including but not limited to Sarah Heimlich, Eric Turner, Shahrab Izadi, Sean Fanello, Danhang Tang, Stephanie Debats, Walter Korman, and Anne Menini.

ACKNOWLEDGMENTS

REFERENCES

5 CONCLUSION

In this demonstration, we present Rapsai, a visual programming platform that enables ML practitioners to interactively explore data augmentation strategies, compare models, and prototype multimedia machine learning (ML) applications. Rapsai provides an efficient means for selecting the most suitable model for a given task and facilitates effective communication of the strengths and limitations of models through the use of examples. We showcase the platform's unique data augmentation and qualitative comparison capabilities, which can facilitate novel approaches for testing model robustness and sharing results in academic publications and presentations. As future work, we plan to extend Rapsai’s capabilities to support text and 3D data and integrate it more closely with the ML training pipeline and cloud-hosted models.