Deep Learning Tools for Audacity: Helping Researchers Expand the Artist's Toolkit

Hugo Flores Garcia[1], **Aldo Aguilar**[1], **Ethan Manilow**[1], **Dmitry Vedenko**[2], **Bryan Pardo**[1]
[1] Northwestern University, [2] Audacity Team
hugofg@u.northwestern.edu

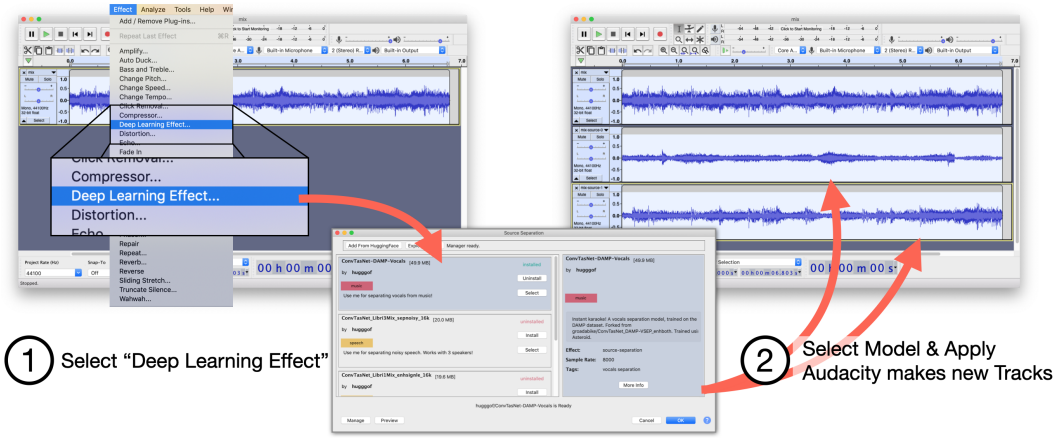


Figure 1: Extracting the vocals for later remixing using a deep source separation model in the Audacity Digital Audio Workstation (DAW). The sound artist selects the desired model available on HuggingFace. It automatically downloads and runs locally, removing the need to upload potentially private audio to a distant website or compile and run code to use a model locally.

1 Introduction

Digital Audio Workstations (DAWs) such as Audacity, ProTools and GarageBand are the primary platforms used in the recording, editing, mixing, and producing of sound art, including music. Making deep models for audio processing easily available on these platforms would be transformative for many artists. For example, remix artists (e.g. Public Enemy, Girl Talk, Kanye West) recombine elements of existing tracks into new works. This often requires spending many hours using a DAW to hand-separate individual vocals, drum beats, or other sounds out of recordings that contain multiple concurrent sounds. Deep learning researchers have created many effective models [9, 7, 6] for separating out voice or individual instruments from music recordings, but these models are typically available only as source code on PapersWithCode or HuggingFace. Artists could save many hours of work, if only there were an easy way to add them into a DAW. More generally, deep models have greatly increased the set of audio generation and manipulation tools that are possible, including some that would be unimaginable or infeasible with traditional DSP techniques, such as melody co-creation [4], automated upmixing [5] and replacing the timbre of one sound (e.g. human voice) in an existing audio recording with that of another (e.g. a violin) [8, 3]. How can we best put these new tools in the hands of the artists?

The standardization of modular effects (e.g. reverberation, equalization, compression) and synthesizer "plugins" has allowed independent developers of traditional DSP-based effects to easily add to the set of tools available to sound artists usind DAWs. Unfortunately, the skills a person needs to develop a

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

novel deep learning model are different than those required to build a plugin for a DAW. In practice, this means that deep learning technology rarely makes its way into DAW-hosted plugins. Furthermore, while it has become increasingly common for deep learning developers to provide runnable source code or even web demos of their work, most end-users do not have the skills to run open-sourced code. Even for those able to overcome this barrier, leaving the DAW to use research code or online demos can form a significant impediment to the workflow, discouraging use. Simplifying the deployment pipeline so that deep model developers can easily make models available to be used directly in the DAW could have a transformative impact on the range of creative tools available to sound artists, producers and musicians.

In this work, we describe a software framework that lets ML developers easily integrate new deep-models into Audacity, a free and open-source DAW that has logged over 100 million downloads since 2015 [1]. Developers upload their trained PyTorch [10] model to HuggingFace's Model Hub. The model becomes accessible through Audacity's UI and loads in a manner similar to traditional plugins. This lets deep learning practitioners put tools in the hands of the artistic creators without the need to do DAW-specific development work, without having to learn how to create a VST plugin, and without having to maintain a server to deploy their models.

2 Model Lifecycle – From Contributor to End User

Our framework defines an API for two common deep learning-based audio paradigms, namely audio-in-audio-out tasks (such as source separation, voice conversion, style transfer, amplifier emulation, etc.) and audio-to-labels tasks (e.g. pitch-tracking, voice transcription, sound object labeling, music tagging). We illustrate the utility of this by incorporating two example deep models into Audacity: a source separator and a musical instrument labeler (see the Appendix). These deep learning plugins are situated alongside traditional, DSP-based effects in the plugin drop down menu, providing a means for an end-user to run deep learning models locally without leaving Audacity (Figure 1).

The model contribution process only requires developers to be familiar with PyTorch. Finished models are serialized with `torchscript`[1] such that they can be run on an end-user's machine. While `torchscript` provides a more limited functionality than full PyTorch, we find that it nonetheless provides most necessary utilities for serializing Audacity-bound models. Serialized models can then be uploaded to the HuggingFace[2] model hub. Models uploaded to HuggingFace with the special `audacity` tag will automatically be scraped, so that end-users can download and run the models through the Audacity interface (shown in Figure 2). Further developer details are provided in the Appendix.

Once a model has been successfully uploaded to HuggingFace, an Audacity user can find it by selecting "Deep Learning Effect" in the "Effect" menu (for audio-in-audio-out models), or "Deep Learning Analyzer" in the "Analyze" menu (for audio-to-label models). When the user selects one of these, a dialog box is shown that is populated with model metadata pulled from HuggingFace. This dialog box gives users the ability to browse models, download a desired model, and run it on selected tracks. The output of the model is either additional audio tracks (updates do not happen in-place) or audio labels, depending on the model selected.

3 Conclusion

This work has the potential to be transformational in increasing an artist's ability to access machine learning models for media creation. Researchers building the next generation of machine learning models for media creation will be able to put these models directly into the hands of the artists. We hope that this work will foster a virtuous feedback loop between model builders and the artists that apply these models to create sound art. The ability to quickly share models between model builders and artists should encourage a conversation that will deepen the work of both groups.

More details are provided in the Appendix and documentation for model contributors[3].

[1] `https://pytorch.org/docs/stable/jit.html`

[2] `https://huggingface.co/`

[3] `https://interactiveaudiolab.github.io/project/audacity`

Ethical Implications of this Work

A small number of deep learning audio effects are available via web-demo or server-side upload. Those with privacy concerns about source material (e.g. artists working on unreleased works) may not be comfortable uploading their audio to a third party. Providing an easy way for artists to use models locally removes this concern, democratizing deep learning tools for audio manipulation in end-user devices. This work enables access to such tools for users who would otherwise have difficulty accessing them. As such, the tools described in this paper could foster an ecosystem of artists and deep learning practitioners. Importantly, this work also has the potential side effect of facilitating misuse by bad actors (e.g. putting tools that facilitate manipulating recorded speech in the hands of those that may wish to falsify evidence). Some may argue that access to deep audio manipulation tools should, therefore, be tightly controlled. We argue that the right solution is not to restrict artist access to any new editing tools that may arise. Instead, we encourage the exploration of solutions for detecting and addressing the unethical use of audio manipulation tools while preserving the open, unrestricted access to these systems for creative expression.

Acknowledgements

This work was funded in part by a Google Summer of Code grant and by NSF Award Number 1901456. We would also like to thank Jouni Helminen and Prem Seetharaman for fruitful conversations.

References

[1] Fosshub.com Audacity Download. `https://www.fosshub.com/Audacity.html`. Accessed: 2020-09-08.

[2] A. Défossez, N. Usunier, L. Bottou, and F. Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

[3] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts. Ddsp: Differentiable digital signal processing. In *International Conference on Learning Representations*, 2020.

[4] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai. AI Song Contest: Human-AI Co-Creation in Songwriting. In *21st International Society for Music Information Retrieval Conference, ISMIR 2020*, pages 708–716. International Society for Music Information Retrieval, 2020.

[5] J. Jarnow. How audio pros 'upmix' vintage tracks and give them new life. `https://www.wired.com/story/upmixing-audio-recordings-artificial-intelligence/`. Accessed: 2021-09-09.

[6] E. Manilow, P. Seetharaman, and B. Pardo. The Northwestern University Source Separation Library. Proceedings of the 19th International Society of Music Information Retrieval Conference (ISMIR 2018), Paris, France, September 23-27, 2018.

[7] E. Manilow, P. Seetharman, and J. Salamon. *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, 2020.

[8] N. Mor, L. Wolf, A. Polyak, and Y. Taigman. Autoencoder-based Music Translation. In *International Conference on Learning Representations*, 2019.

[9] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stöter, M. Hu, J. M. Martín-Doñas, D. Ditter, A. Frank, A. Deleforge, and E. Vincent. Asteroid: the PyTorch-based audio source separation toolkit for researchers. In *Proc. Interspeech*, 2020.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Appendix

A Examples

We provide built-in models for several use cases: source separation (isolating vocals, or musical instruments from a recording), voice enhancement, and musical instrument recognition. In our supplementary video material [4], we showcase the creative opportunities allowed by our proposed framework by walking through the process of remixing a pop song, using different source separation models.

B End user workflow

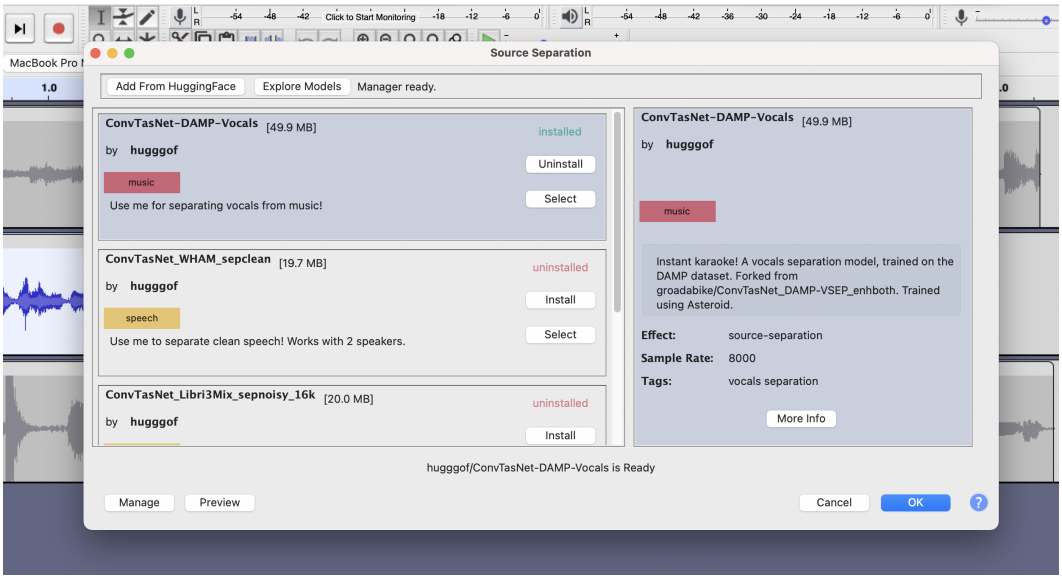
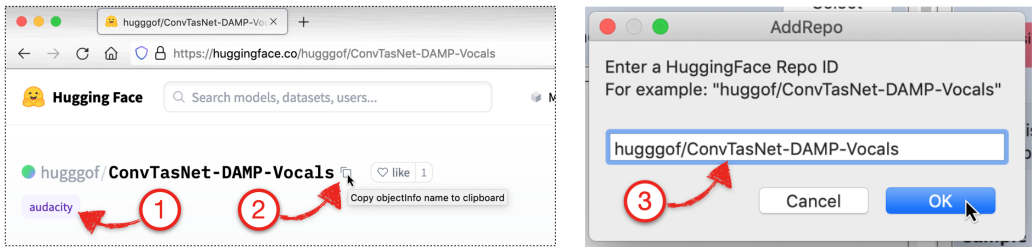


Figure 2: Model manager UI in Audacity. Users can explore a curated set of high-quality neural network models available on HuggingFace for a variety of use cases. Models can be contributed by anyone in the community. To see a full list of all models, the user clicks on "Explore Models", on the top left corner of the window.

From an end-user perspective, Audacity offers two kinds of deep-learning tools, one for processing waveform-to-waveform (i.e. "Deep Learning Effect") and one for processing waveform-to-labels (i.e. "Deep Learning Analyzer"). Users can process their audio by selecting a target audio region, followed by opening their desired deep learning tool. Deep Learning tools can be accessed via the "Effect" menu on Audacity's menu bar, while Analyzer tools can be accessed via the "Analyze" menu. After installing and selecting a model, a user can apply the effect by pressing the "OK" button on the lower right, similar to other built-in offline effects in Audacity. Unlike other Audacity effects, however, the output audio is written to new tracks, instead of in-place.

When a user opens either Deep Learning Effect or Analyzer for the first time, they are shown the model manager interface, shown in Figure 2. The model manager displays a curated list of repositories from HuggingFace, filtered by type (i.e. "Effect" or "Analyzer"). The model manager interface resembles a package manager interface, providing a way to navigate through the list and install a model with a single button press. Additionally, users can click on the "Explore Models" button on the top toolbar to open a HuggingFace search query in an external browser for viewing/installing all models contributed to HuggingFace by the Audacity community. The user can add an uncurated model to their local collection by copying the repository's ID (e.g. `hugggof/ConvTasNet-Vocals`) into the "Add from HuggingFace" dialog, shown in Figure 3b.

[4] `https://interactiveaudiolab.github.io/project/audacity`



(a) (b)

Figure 3: Figure 3a (left) shows an annotated screenshot from the HuggingFace website showing a model that can be run with Audacity. For a model creator to make their model accessible to Audacity users, the model repository must have the `audacity` tag, denoted by the red "1" on the bottom left of Fig. 3a. From the HuggingFace website, users can click on the copy icon next to the repo's ID (denoted by the red "2") to copy the ID onto their clipboard. To add the model to Audacity, the user must paste the repo's ID on the dialog box shown on the right in Figure 3b (shown by the red "3").

Although the deep learning effects proposed here are applied off-line (i.e., not real-time), we find anecdotally that large models can perform inference quite quickly, even on commodity hardware. As an illustrative example, we used a mid-2014 MacBook Pro (7 years old at the time of this writing) to run the large source separation network Demucs [2] ($\approx$ 265M parameters) on 210 sec (3:30 min) of monophonic audio recorded at a sample rate of 44.1 kHz (compact-disc quality audio). It took 80 seconds to process. This speed is in line with advanced DSP audio processing commonly applied by DAW users. Given that, we believe this delay is acceptable for end-users, and this opens up the possibility of developing real-time deep learning effects in future work.

C Contributing models to Audacity

Deep learning models for Audacity are hosted in the HuggingFace model hub. The model contribution process is meant to be as straightforward as possible, requiring model contributors to be familiar with only PyTorch [10], a popular open-source and python-based deep learning framework.

In order to make a PyTorch model compatible with our framework, a model contributor must 1) serialize their model into a `torchscript` file, 2) create a metadata file for the model, and 3) upload their model to the HuggingFace model hub. We provide an outline of the model contribution process in this work, as well as make thorough documentation for model contributors available online[5]. Additionally, we provide an example notebook[6] that serializes a source separation model pre-trained using the Asteroid library [9].

C.1 Writing and Exporting Models for Audacity

After training a model using PyTorch, a contributor must serialize their model into the `torchscript` format to use in Audacity. We note that this poses limitations not present in pure Python environments, because the supported operations in `torchscript` are a subset of the wider Python and PyTorch operation set. Nevertheless, we find that most necessary utilities for building end-to-end audio models are `torchscript` compatible: `torchscript` supports a large set of built-in functions and data types [7], and the `torchaudio` package contains many serializable pre- and postprocessing audio modules (e.g., `Spectrogram`, `GriffinLim`, `MelSpectrogram`, Mel Filter Banks, etc.).

C.2 Model Architectures

As of this work, the deep learning framework for Audacity supports two kinds of deep learning-based effects: 1) waveform-to-waveform, and 2) waveform-to-labels. Both effects are **agnostic** to

[5] `https://github.com/hugofloresgarcia/audacitorch`

[6] `https://github.com/hugofloresgarcia/audacitorch/blob/main/notebooks/example.ipynb`

[7] `https://pytorch.org/docs/stable/jit_builtin_functions.html`

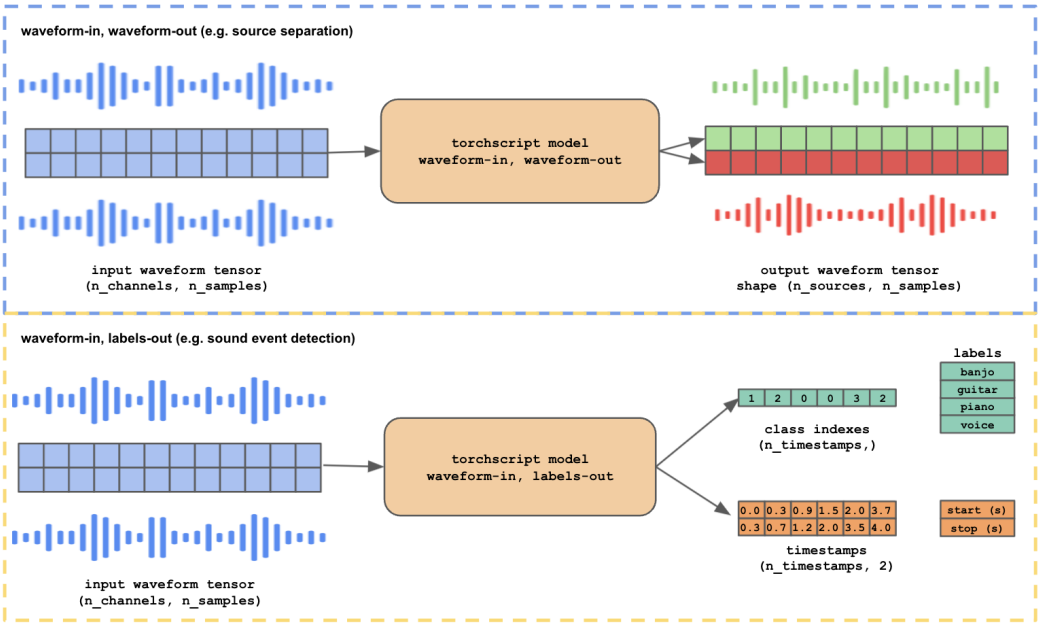


Figure 4: Tensor graphs for Audacity models: 1) waveform-to-waveform (top), and 2) waveform-to-labels (bottom). Note that the framework is agnostic to the internal model architecture, as long as it satisfies the input-output constraints, shown above. The `timestamps` tensor is shown transposed for compactness.

the internal model architecture, assuming the model is serializable via `torchscript` and satisfies Audacity's input-output constraints. These constraints are shown in Figure 4, and outlined below:

- During inference, both waveform-to-waveform and waveform-to-labels models are passed a raw audio multichannel waveform tensor with shape `(num_input_channels, num_samples)` as input to the forward pass.
- Waveform-to-waveform effects receive a single multichannel audio track as input, and may write to a variable number of new audio tracks as output. Example models for waveform-to-waveform effects include source separation, neural upsampling, guitar amplifier emulation, generative models, etc. Output tensors for waveform-to-waveform models must be multichannel waveform tensors with shape `(num_output_channels, num_samples)`. For every audio waveform in the output tensor, a new audio track is created in the Audacity project.
- Waveform-to-labels receive a single multichannel audio track as input, and may write to a single label track as output. The waveform-to-labels effect can be used for many audio analysis applications, such as voice activity detection, sound event detection, musical instrument recognition, automatic speech recognition, etc. The output for waveform-to-labels models must be a tuple of two tensors. The first tensor corresponds to the class indexes for each label present in the waveform, shape `(num_timesteps,)`. The second tensor must contain timestamps with start and stop times for each label, shape `(num_timesteps, 2)`.

C.3 Model Metadata

Certain details about the model, such as its sample rate, tool type (e.g. waveform-to-waveform or waveform-to-labels), list of labels, etc. must be provided by the model contributor in a separate `metadata.json` file. In order to help users choose the correct model for their required task, model contributors are asked to provide a short and long description of the model, the target domain of the model (e.g. speech, music, environmental, etc.), as well as a list of tags or keywords as part of the metadata.

C.4 Uploading to HuggingFace

Once a model has been serialized to `torchscript` and a metadata file has been created, model contributors can upload their models to repositories in the HuggingFace model hub. If the tag `audacity` is included in the repository's `README.md`, the repository will be visible to Audacity users via the "Explore HuggingFace" button (shown in Figure 2).