# welcome

Welcome! The Distant Viewing Explorer is a free, open-source, web-based software designed to help anyone apply state-of-the-art AI algorithms to study visual and multimodal collections. While there has been a rapid increase in the public availability and capability of AI models over the past decade, using these advances directly without extensive programming and testing can be challenging. The goal of this software is to provide a way of applying a set of curated computer vision models directly to any dataset of interest while also providing a pathway for working with larger datasets and additional models using additional tools and resources.

We currently support 15 different models that address various needs and data types. All of the models have been trained on contemporary datasets, with the image-based models trained entirely on digital photography. While some models generalize well to historical data and other formats, in many cases, they do not work as well as expected. We have included a collection of several example datasets to illustrate the potential benefits and challenges of each of the proposed approaches.

No coding is needed to run the Distant Viewing Explorer. The code runs directly through the graphical interface. The software runs entirely in the browser on your machine. No data is being shared with us or anyone else. To learn how to get started, follow the links on the left.

We hope that you find the site interesting and helpful. Please contact us directly if you have any questions or comments.

# started

The Distant Viewing Explorer can be used directly in your browser without additional setup or logins. Each available model can be selected using the menu on the left. The models are grouped by the type of data they work with. While our focus is applications to still and moving images, we include several models working with text that are useful for studying associated textual data, such as subtitles, reviews, and image captions. Feel free to go through the models in order or jump around to those you are interested in.

Each model page is organized in a consistent format to make it easier to use and navigate. For each model, we have a short description of what it is designed to do. A Learn More tag above the description toggles a longer discussion of the model, along with citations and possible applications. Below the description, four buttons are shown with only the “Load Model” button initially available. Clicking on this will download the model files. The first time downloading the files may take a minute or two to finish, with subsequent loading usually faster through cached model versions. The model load progress can be seen in the progress directly below the buttons.

After the model has loaded, buttons for selecting either an example dataset or uploading your own data are shown. The load example (a good place to start) will show a page with several example datasets you can select. The text-based models also allow you to enter your input directly through the input box on the right-hand side of the screen. The image collections have a smaller set and a larger one to choose from. When loading your images, you can select a set of images within the same directly. After selecting the inputs, the model will run, with progress shown in the progress bar. For video inputs, the video will start playing as the model runs. The results will be visible on the right when the model is finished. Finally, a download button will become active to save the model results locally to your machine. The download format will depend on the model selected.

The Embedding and Zero-Shot models feature an additional layer of interactivity, allowing users to further search and explore the results after they are produced. Instructions for these are given on the individual pages.

If you find a specific model useful and want to learn more about how to apply and adapt it in a coding environment, the Python button under the model name will open a page with additional code.

# citation

If you use the Distant Viewing Explorer for your work, please cite it as:

# moreinfo

For more information about the techniques here, we suggest the following resources for further information about the AI models and their applications within the humanities:

* [Distant Viewing Lab](https://distantviewing.org)
* [Humanities Data in R](https://humanitiesdata.org/)
* [AudioVisual in Digital Humanities (AVinDH) SIG](https://avindh.org/)
* [Hugging Face](https://huggingface.co/)
* [AudioVisual Data in DH: Special Issue of DHQ](http://digitalhumanities.org/dhq/vol/15/1/index.html)

# classify

## short

Image classification aims to identify a category from a pre-selected set describing an image. Outputs provide the top five most likely categories predicted by the model, along with probability scores indicating the model’s confidence. Image classification is one of the earliest tasks in computer vision analysis, so it is a good place to start. Try this with the ILSVRC example dataset, as it resembles the data used to train these models more closely.

## long

Image classification aims to identify a category from a pre-selected set describing an image. Outputs provide the top five most likely categories predicted by the model, along with probability scores indicating the model’s confidence. Image classification is one of the earliest tasks in computer vision analysis, so it is a good place to start. Try this with the ILSVRC example dataset, as it resembles the data used to train these models more closely.

### References

@inproceedings{he2016deep,

title={Deep residual learning for image recognition},

author={He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian},

booktitle={Proceedings of the IEEE conference on computer vision and pattern recognition},

pages={770--778},

year={2016}

}

# object

## short

Object detection aims to find, and usually locate with a box, objects within an image that come from a pre-determined set of classes. The Microsoft COCO example dataset gives a good idea of the kind of data the model was trained with. Try to test it with various inputs and see how well (or not) it does!

## long

Object detection aims to find, and usually locate with a box, objects within an image that come from a pre-determined set of classes. The Microsoft COCO example dataset gives a good idea of the kind of data the model was trained with. Try to test it with various inputs and see how well (or not) it does!

### References

@article{DBLP:journals/corr/abs-2005-12872,

author = {Nicolas Carion and

Francisco Massa and

Gabriel Synnaeve and

Nicolas Usunier and

Alexander Kirillov and

Sergey Zagoruyko},

title = {End-to-End Object Detection with Transformers},

journal = {CoRR},

volume = {abs/2005.12872},

year = {2020},

url = {https://arxiv.org/abs/2005.12872},

archivePrefix = {arXiv},

eprint = {2005.12872},

timestamp = {Thu, 28 May 2020 17:38:09 +0200},

biburl = {https://dblp.org/rec/journals/corr/abs-2005-12872.bib},

bibsource = {dblp computer science bibliography, https://dblp.org}

}

# depth

## short

For photography, depth estimation identifies the relative distance of each pixel in the image away from the camera. We can extend this to other types of images by considering the objects' depths from the artist's perspective. The output will show the closest pixels in white and the farthest pixels in black. Hoving over the production reveals the original image for comparison. Downloading the results will provide a zipped directory of all the depth-predicted images.

## long

For photography, depth estimation identifies the relative distance of each pixel in the image away from the camera. We can extend this to other types of images by considering the objects' depths from the artist's perspective. The output will show the closest pixels in white and the farthest pixels in black. Hoving over the production reveals the original image for comparison. Downloading the results will provide a zipped directory of all the depth-predicted images.

### References

@misc{yang2024depth,

title={Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data},

author={Lihe Yang and Bingyi Kang and Zilong Huang and Xiaogang Xu and Jiashi Feng and Hengshuang Zhao},

year={2024},

eprint={2401.10891},

archivePrefix={arXiv},

primaryClass={cs.CV}

}

# segment

## short

Image segmentation attempts to identify what things or elements are associated with every pixel of an image. These can be either objects (such as with object detection) or regions of “stuff” such as the sky, water, or grass. When the output is produced, hovering over the image will show the predicted identity of each group of pixels. The overall proportions for each category are shown at the bottom of the images.

## long

Image segmentation attempts to identify what things or elements are associated with every pixel of an image. These can be either objects (such as with object detection) or regions of "stuff," such as the sky, water, or grass. When the output is produced, hovering over the image will show the predicted identity of each group of pixels. The overall proportions for each category are shown at the bottom of the images.

### References

@article{DBLP:journals/corr/abs-2105-15203,

author = {Enze Xie and

Wenhai Wang and

Zhiding Yu and

Anima Anandkumar and

Jose M. Alvarez and

Ping Luo},

title = {SegFormer: Simple and Efficient Design for Semantic Segmentation with

Transformers},

journal = {CoRR},

volume = {abs/2105.15203},

year = {2021},

url = {https://arxiv.org/abs/2105.15203},

eprinttype = {arXiv},

eprint = {2105.15203},

timestamp = {Wed, 02 Jun 2021 11:46:42 +0200},

biburl = {https://dblp.org/rec/journals/corr/abs-2105-15203.bib},

bibsource = {dblp computer science bibliography, https://dblp.org}

}

# embed

## short

Image embeddings are a way to find similarities between two pairs of images (they also have other applications, as described in the ‘Learn More’ link above). Here, we will run an embedding over a set of images. Then, the visualization on the right will allow you to click on an image and sort the remainder of the collection based on similarity to the selected image. Scores at the bottom range from 0 (no similarity) to 100 (identical). Try any of the large example datasets; they have pre-computed embedings making them fast and easy to experiment with.

## long

Image embeddings are a way to find similarities between pairs of images (they also have other applications, as described in the ‘Learn More’ link above). Here, we will run an embedding over a set of images. Then, the visualization on the right will allow you to click on an image and sort the remainder of the collection based on similarity to the selected image. Scores at the bottom range from 0 (no similarity) to 100 (identical). Try any of the large example datasets; they have pre-computed embedings making them fast and easy to experiment with.

### References

@misc{wu2020visual,

title={Visual Transformers: Token-based Image Representation and Processing for Computer Vision},

author={Bichen Wu and Chenfeng Xu and Xiaoliang Dai and Alvin Wan and Peizhao Zhang and Zhicheng Yan and Masayoshi Tomizuka and Joseph Gonzalez and Kurt Keutzer and Peter Vajda},

year={2020},

eprint={2006.03677},

archivePrefix={arXiv},

primaryClass={cs.CV}

}

# shotboundary

## short

Shot boundary detection is the task of identifying where a video transitions from one shot to another. Transitions can be a hard cut (the most common type, showing a sudden break from one shot to the next) or a transition shot such as a fade-in or cross-fade. The algorithm used in the explorer is simple, taking the differences in pixels from one frame to the next. More accurate methods, which have not yet been ported to the web, are shown in the Python code. Downloading the results will provide a zip file with one frame from each shot. You can use these images in any of the image-based annotations above.

## long

Shot boundary detection is the task of identifying where a video transitions from one shot to another. Transitions can be a hard cut (the most common type, showing a sudden break from one shot to the next) or a transition shot such as a fade-in or cross-fade. The algorithm used in the explorer is simple, taking the differences in pixels from one frame to the next. More accurate methods, which have not yet been ported to the web, are shown in the Python code. Downloading the results will provide a zip file with one frame from each shot. You can use these images in any of the image-based

### References

@article{soucek2020transnetv2,

title={TransNet V2: An effective deep network architecture for fast shot transition detection},

author={Sou{\v{c}}ek, Tom{\'a}{\v{s}} and Loko{\v{c}}, Jakub},

year={2020},

journal={arXiv preprint arXiv:2008.04838},

}

# transcription

## short

Transcription, also known as speech-to-text, takes an audio file and converts it into a machine-readable text of the words being spoken. Most models, as is the one here, are language-dependent. You must select your desired language from the menu below before getting started. The model output provides a timestamp for where a word is spoken. Clicking on a word in the transcript will jump the video to the time stamp when it predicts the word being spoken.

## long

Transcription, also known as speech-to-text, takes an audio file and converts it into a machine-readable text of the words being spoken. Most models, as is the one here, are language-dependent. You must select your desired language from the menu below before getting started. The model output provides a timestamp for where a word is spoken. Clicking on a word in the transcript will jump the video to the time stamp when it predicts the word being spoken.

### References

@misc{radford2022whisper,

doi = {10.48550/ARXIV.2212.04356},

url = {https://arxiv.org/abs/2212.04356},

author = {Radford, Alec and Kim, Jong Wook and Xu, Tao and Brockman, Greg and McLeavey, Christine and Sutskever, Ilya},

title = {Robust Speech Recognition via Large-Scale Weak Supervision},

publisher = {arXiv},

year = {2022},

copyright = {arXiv.org perpetual, non-exclusive license}

}

# diarization

## short

Diarization is another audio-based annotation. It attempts to identify when different speakers are speaking in an audio recording. It is very useful for studying film and television data, often being paired with the transcription task. Note that the model used here begins to have trouble when the number of speakers is large. The Python code provides some ways of addressing this issue with larger datasets.

## long

Diarization is another audio-based annotation. It attempts to identify when different speakers are speaking in an audio recording. It is very useful for studying film and television data, often being paired with the transcription task. Note that the model used here begins to have trouble when the number of speakers is large. The Python code provides some ways of addressing this issue with larger datasets.

### References

@inproceedings{Plaquet23,

author={Alexis Plaquet and Hervé Bredin},

title={{Powerset multi-class cross entropy loss for neural speaker diarization}},

year=2023,

booktitle={Proc. INTERSPEECH 2023},

}

@inproceedings{Bredin23,

author={Hervé Bredin},

title={{pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe}},

year=2023,

booktitle={Proc. INTERSPEECH 2023},

}

# sentiment

## short

Sentiment analysis is the process of assigning a subjective label to a text indicating the affective state or intention of the writer/speaker. We include it here as it can be helpful for transcriptions and movie/television reviews. The model used here provides a simple positive or negative prediction and a score. Try a variety of example texts and see how the scores align with your understanding of the data.

## long

Sentiment analysis is the process of assigning a subjective label to a text indicating the affective state or intention of the writer/speaker. We include it here as it can be helpful for transcriptions and movie/television reviews. The model used here provides a simple positive or negative prediction and a score. Try a variety of example texts and see how the scores align with your understanding of the data.

### References

@article{sanh2019distilbert,

title={DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv 2019},

author={Sanh, Victor and Debut, L and Chaumond, J and Wolf, T},

journal={arXiv preprint arXiv:1910.01108},

year={2019}

}

# stars

## short

In addition to positive or negative labels, we can also predict more granular categories from textual inputs. For looking at how audiences respond to visual data, it can be helpful to predict sentiment on a scale from 1 to 5, as is often the case in product reviews. Here, we have a model that assigns percentages to each possible number of stars between 1 and 5.

## long

In addition to positive or negative labels, we can also predict more granular categories from textual inputs. For looking at how audiences respond to visual data, it can be helpful to predict sentiment on a scale from 1 to 5, as is often the case in product reviews. Here, we have a model that assigns percentages to each possible number of stars between 1 and 5.

### References

@misc {nlp\_town\_2023,

author = { {NLP Town} },

title = { bert-base-multilingual-uncased-sentiment (Revision edd66ab) },

year = 2023,

url = { https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment },

doi = { 10.57967/hf/1515 },

publisher = { Hugging Face }

}

# toxic

## short

As a final example of tagging text with pre-defined labels, we can also categorize the toxicity of user reviews. This is helpful when working with user-produced responses to artworks, films, or television shows.

## long

As a final example of tagging text with pre-defined labels, we can also categorize the toxicity of user reviews. This is helpful when working with user-produced responses to artworks, films, or television shows.

### References

@misc{Detoxify,

title={Detoxify},

author={Hanu, Laura and {Unitary team}},

howpublished={Github. https://github.com/unitaryai/detoxify},

year={2020}

}

# mask

## short

Mask prediction is a text processing task in which one or more words are masked or hidden, and the model attempts to predict what word(s) should fill the missing gap. The input for this model requires a string with ‘[MASK](#mask)’ somewhere in it. It will output the predicted word to fill in the gap. Try a special masked version of the AFI movie quotes dataset. See the Learn More link above for a description of projects using this for research.

## long

Mask prediction is a text processing task in which one or more words are masked or hidden, and the model attempts to predict what word(s) should fill the missing gap. The input for this model requires a string with ‘[MASK](#mask)’ somewhere in it. It will output the predicted word to fill in the gap. Try a special masked version of the AFI movie quotes dataset. See the Learn More link above for a description of projects using this for research.

### References

@article{DBLP:journals/corr/abs-1810-04805,

author = {Jacob Devlin and

Ming{-}Wei Chang and

Kenton Lee and

Kristina Toutanova},

title = {{BERT:} Pre-training of Deep Bidirectional Transformers for Language

Understanding},

journal = {CoRR},

volume = {abs/1810.04805},

year = {2018},

url = {http://arxiv.org/abs/1810.04805},

archivePrefix = {arXiv},

eprint = {1810.04805},

timestamp = {Tue, 30 Oct 2018 20:39:56 +0100},

biburl = {https://dblp.org/rec/journals/corr/abs-1810-04805.bib},

bibsource = {dblp computer science bibliography, https://dblp.org}

}

# zeroshot

## short

Multimodal zero-shot learning extends and combines the ideas behind object detection and image embedding. The goal is to identify a similarity score between a textual description and an image such that the score is large when the description serves as a good caption for the image. Captions allow us to search for any categories (however complex) without creating a specialized model. Here, we first run an embedding over the entire collection and then, on the right, allow for inputting a search string. We suggest starting with the large example datasets as they have pre-computed embeddings and quickly show the powerful possibilities of this approach.

## long

Multimodal zero-shot learning extends and combines the ideas behind object detection and image embedding. The goal is to identify a similarity score between a textual description and an image such that the score is large when the description serves as a good caption for the image. Captions allow us to search for any categories (however complex) without having to create a specialized model for them. Here, we first run an embedding over the entire collection and then, on the right, allow for inputting a search string. We suggest starting with the large example datasets as they have pre-computed embeddings and quickly show the powerful possibilities of this approach.

### References

@misc{zhai2023sigmoid,

title={Sigmoid Loss for Language Image Pre-Training},

author={Xiaohua Zhai and Basil Mustafa and Alexander Kolesnikov and Lucas Beyer},

year={2023},

eprint={2303.15343},

archivePrefix={arXiv},

primaryClass={cs.CV}

}

# caption

## short

Automatic image captioning is a multimodal technique where we input an image and have the model create a short caption. The small model used in the browser produces interesting results alongside many errors. Try to see how the model does will different types of inputs (paintings, historic images, modern images, etc.). More accurate models that extend the capabilities here are introduced in the Python code.

## long

Automatic image captioning is a multimodal technique where we input an image and have the model create a short caption. The small model used in the browser produces interesting results alongside many errors. Try to see how the model does will different types of inputs (paintings, historic images, modern images, etc.). More accurate models that extend the capabilities here are introduced in the Python code.

### References

@inproceedings{mishra2024image,

title={Image Caption Generation using Vision Transformer and GPT Architecture},

author={Mishra, Swapneel and Seth, Saumya and Jain, Shrishti and Pant, Vasudev and Parikh, Jolly and Jain, Rachna and Islam, Sardar MN},

booktitle={2024 2nd International Conference on Advancement in Computation \& Computer Technologies (InCACCT)},

pages={1--6},

year={2024},

organization={IEEE}

}