Case Study: Contrail Segmentation

On the Importance of Understanding Your Data and Task

Rolls off the tongue, doesn't it? - The Authors of This Notebook

This notebook presents a set of exercises in the form of a case study on segmenting contrails from satellite image (also referred to as remote sensing data).

Prerequisties:

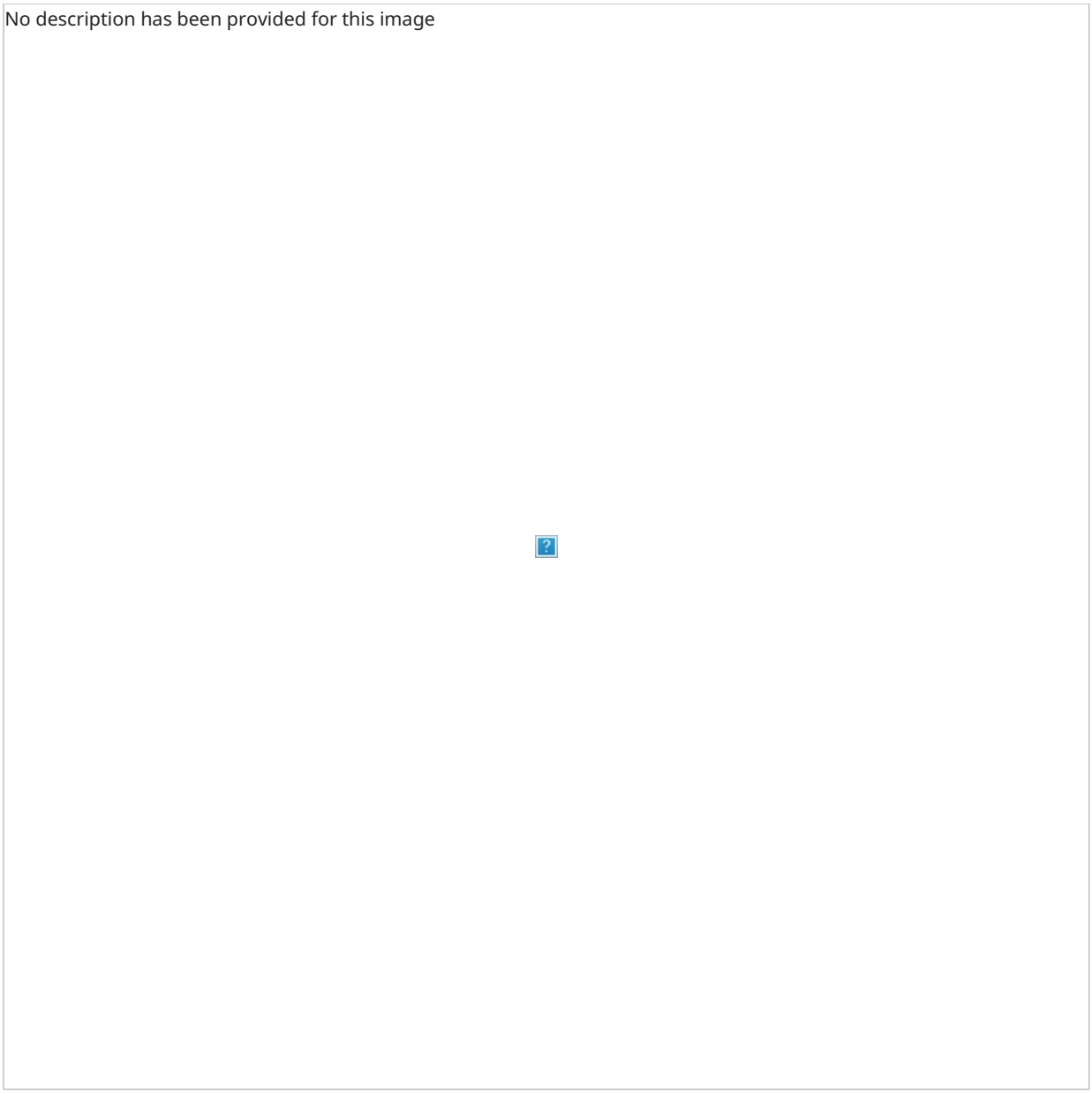
- Basic machine learning knowledge
- Basic familiarity with the paper "Few-Shot Contrail Segmentation in Remote Sensing Imagery With Loss Function in Hough Space" by Junzi et al. (https://ieeexplore.ieee.org/document/10820969). We do not expect you to understand the whole paper, but rather to have an idea of what is being done.

The goal by the end of this notebook is for students to:

- Gain an exposure to a new application of artificial intelligence - extracing semantics from satellite images
- Understand when they can consider the specifics of their data and task
- Gain exposure to methods to take advantage of their task
- Understand how to evaluate machine learning tasks based on the goal they are solving
- Go beyond the technical understanding of the problem and consider where this application will be used, will it even be helpful, etc.

Introduction: Contail Segmentation

The paper "Few-Shot Contrail Segmentation in Remote Sensing Imagery With Loss Function in Hough Space" by Junzi et al. focuses on creating an automatic segmentation procedure for contrails when using few data samples by taking advantage of what we know about the problem. In this section, the problem is introduced from the very basics of what a contrails is, to a mathematical formulation of the issue.



What is a contrail?

Contrails or vapour trails are line-shaped clouds produced by aircraft engine exhaust or changes in air pressure, typically at aircraft cruising altitudes several kilometres/miles above the Earth's surface. (Definition - https://en.wikipedia.org/wiki/Contrail)

Why does it matter?

The Sun emits solar radiation towards the Earth that the ground traditionally reflects back. However, the formation of the ice crystals in contrails creates a dense enough "shield" to reflect a part of them back. This impacts the amount of radiation trapped around the Earth and thus contributes to the temperature. They also have a potential cooling effect on sun rays getting reflected back towards the Sun, though this effect is currently estimated to be smaller.

Satellite Image

Satellite images are a unique form of data. As the name implies, they are taken by a satellite, which leads to charactersitics about it, such as:

- **top-down**: the perspective of the image is always orthogonal towards the ground
- **distance from the Earth**: Depending on the image, and satellite both the distance to the Earth and the resolution of the image is different.
  - For this reason, the resolution is measured in terms of actual distance (e.g. A resolution of 30cm means that 30cm of information is encoded per pixel.)
  - Common resolutions are 30cm, 1m, 2m. Correspondingly a smaller distance corresponds to higher quality, more expense, harder to get, etc.
- **multispectral data**: Sensory information from satellites is more advanced than traditional cameras. They can capture more than just the color spectrum. Each one is a special band (channel in images). Below are listed some examples:
  - RGB (Red-Green-Blue): The channels traditional cameras capture, and are blended together to create a final image
  - Infrared/Near-Infrared: A channel where some surfaces react differently to - for example, vegetation reacts differently to infrared light. Can reveal new information, not visible to the naked eye.
  - Point Clouds (LiDAR): A channel capturing distance to the earth. Used to map out things like elevation via the laser reflecting back.

- And many others, depending on the sensor used ... We will introduce them in this notebook if and as necessary.

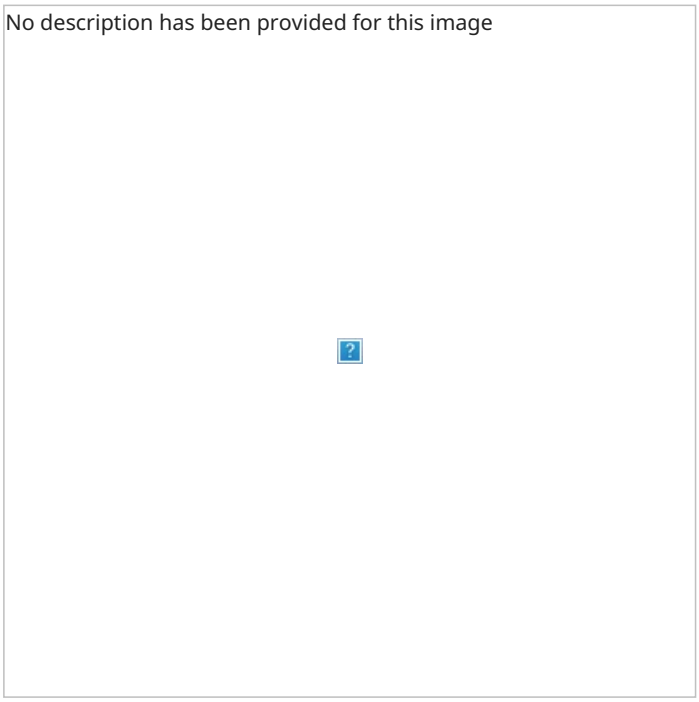


Image Source: GeeksForGeeks

Below are listed some example images of contrail images taken from different satellites and bands:

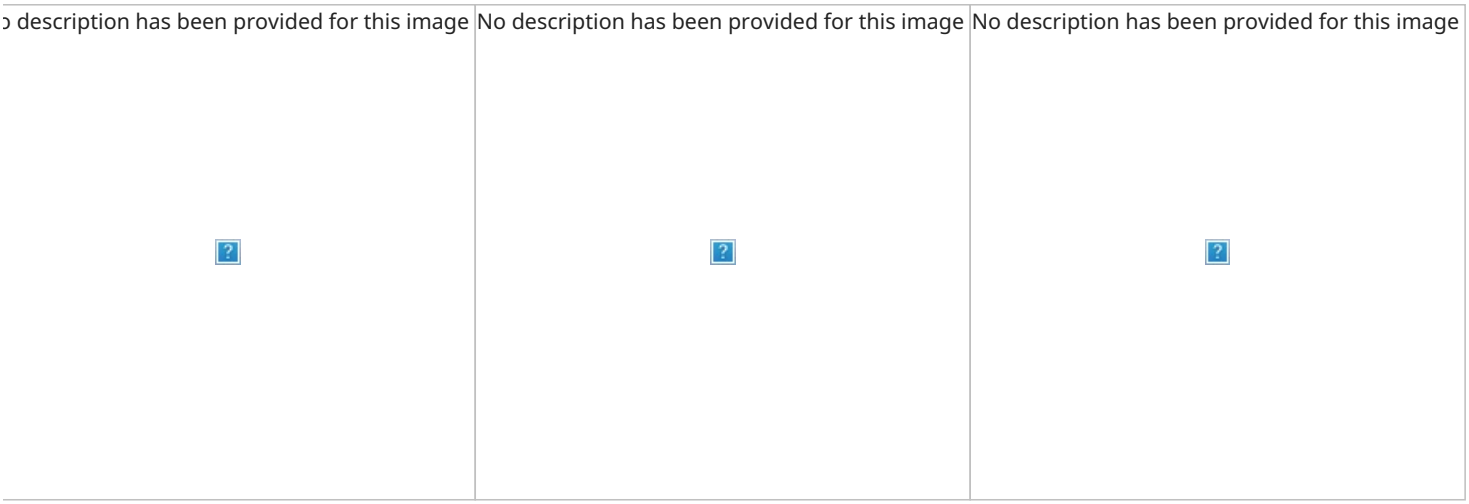


Image Source TBD

Project Setup

IF LOCAL: Create an Anaconda Environment and Download Requirements

If you haven't already setup the codebase following the instructions of the README, you can do so here. Otherwise, you can skip running this.

IF ON COLAB: Download Repository

Get Imports

What is Contrail Segmentation?

What are we trying to accomplish? From the name **contrail segmentation**, and the information beforehand, we know that we are trying to extract information out of the satellite images. To do so, we need to define **segmentation** first.

Segmentation

Recall **classification**, abstracted from any machine learning model. Given an input x, we want to get an output $\widehat y$ corresponding to the correct class y of the object x.

In remote sensing, our inputs are images. Therefore $x \in \mathbb R^{W \times H \times C} $ where W, H are the height and width (in pixels) of the current image and C the number of channels total the image (recall RGB or Infrared introduced previously). Mathematically, this is a **tensor** (it is not very important for today, but good to know in general).

In **segmentation**, we want to semantically understand our image. A way to do this is to somehow classify what we are seeing in our image. One way to do this is to output another image, classifying each pixel, whether it belongs to the object/s we are looking for. This can be defined as creating a function $f:x \in \mathbb R^{W \times H \times C} \to y \in \mathbb Z_n$ where n is the number of classes we have in our image. An illustration is provided below. Specifically, this is a case called **semantic segmentation**.

No description has been provided for this image

Image Source

The case of **contrail segmentation** can then be examined as just a subset of **semantic segmentation** where our classes are $n=2$ - "contrail" and "background". Illustration of x and y below:

No description has been provided for this image	No description has been provided for this image

Image Source TBD

This is a challenging task. Throughout the study, the authors present ways they try to compensate for their lack of data (only around 30 images). **This is a common occurence in the ML industry**. In the following sections, you will go through exercises examining each of their approaches, try to add to them and apply them to a different case, and reason about if this is a correct application.

Data Scarcity - Data Augmentation

Scarcity is an issue everybody who works with data faces eventually, even in foundational work - https://www.eecis.udel.edu/~shatkay/Course/papers/NetworksAndCNNClasifiersIntroVapnik95.pdf. **Data Augmentation** is a way to reduce overfitting on incomplete datasets.

Intuitively, it introduces distortions to the small data samples, which vary during training so that the machine learning model of choice does not focus on arbitrary aspects of the data (such as the position of the object, such as a specific position or number).

Mathematically, it changes the sampling from the training data set X_{train} by a affine transformation function $g: \mathbb R^n \to \mathbb R^n $, which is applied on every sample point $x \in X$ and outputs a different transformation each time. This change effetively increases the diversity in the training set distribution and brings the final estimate closer to the maximum likelihood on the complete data set X where $X_{train} \subseteq X$.

Example

Imagine a scenario where we have a small training set on flowers (such as the Iris dataset), where all daisies and poppies in the training set have a sepal width s of 5 and 7 centimeters, respectively. The test set, and correspondingly all real examples may have no such examples and there are daisies and poppies of many more sizes. A data augmentations technique may be making g apply noise to the inputs, putting the daisies and poppies's s within some standard deviation giving a random variable over the dataset $$s_{\text{augmented}} = s + \varepsilon, \quad \varepsilon \sim \mathcal N(0, \sigma^2)$$

For instance, daisies might be sampled with sepal widths 5 ± 0.5.

This section introduces data augmentation in two modalities: audio and images. Later, we will also show the value of that on our task of contrails.

One is a fully-guided example, the other ones need to have either code filled in and/or theoretical questions answered. Finally, this is all related to the contrail case.

Audio

Here we show a simple task: audio classification. Given an audio file, can we understand automatically what number is being said based on the recording? For this purpose, we are going to use a subset of the AudioMNIST dataset, and evaluate performance on it both with and without data augmentation.

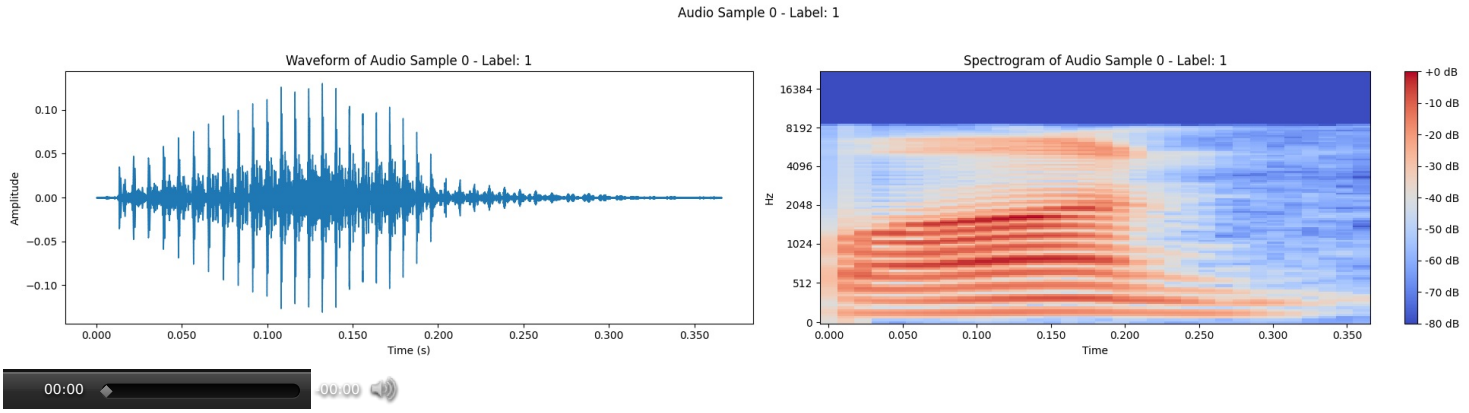
```
Cloning into 'free-spoken-digit-dataset'...
remote: Enumerating objects: 4260, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (40/40), done.
remote: Total 4260 (delta 25), reused 8 (delta 8), pack-reused 4212 (from 1)
Receiving objects: 100% (4260/4260), 30.38 MiB | 21.32 MiB/s, done.
Resolving deltas: 100% (129/129), done.
```

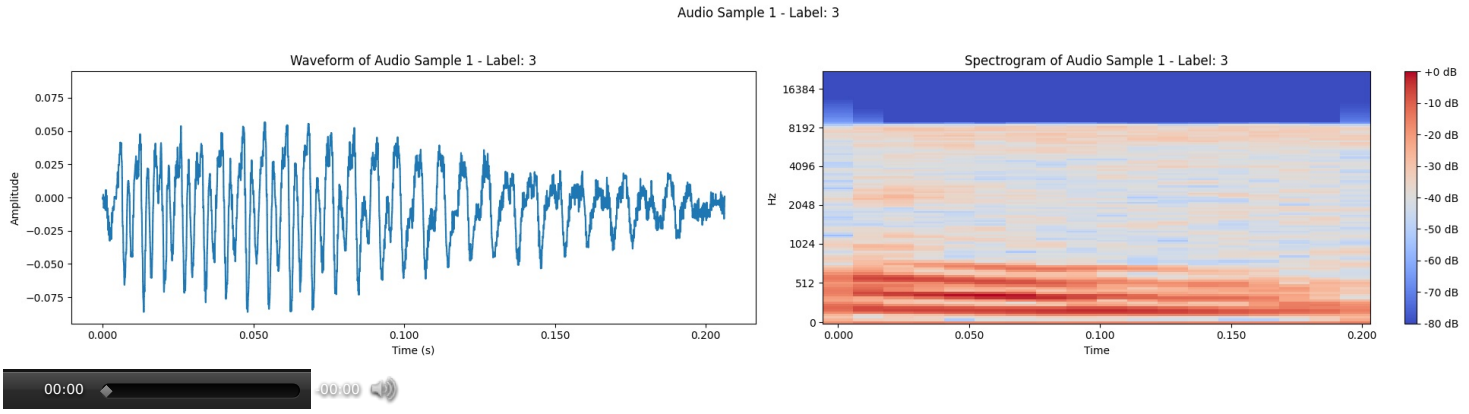
https://github.com/soerenab/AudioMNIST - AudioMNIST

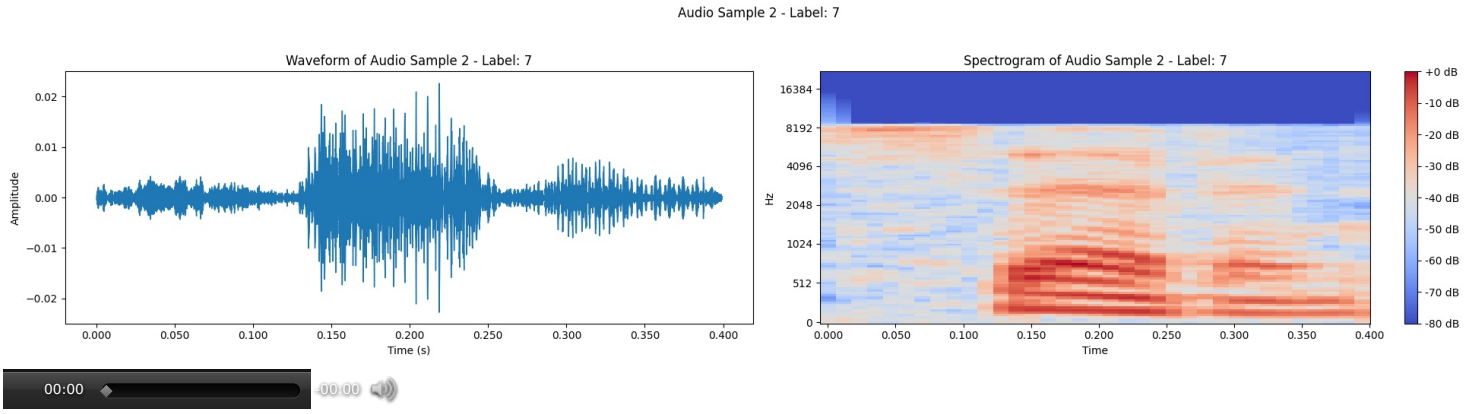
```
100%|██████████| 1000/1000 [00:29<00:00, 34.20it/s]
```

Original Audio Samples

An audio file is created when an analog processor converts sound waves into a sequence of amplitudes and spectrograms. They respectively give an idea of when the volume and frequency are higher. When it is played, it is also just an array of amplitudes, showing the intensity. By looking at the waveform graphs, you can clearly discern when someone is speaking, and also play the audio.







Augmented Audio Samples

This signal seems to have lots of amplitudes, right, and varying length, based on the person, voice, etc. We have prepared an MNIST dataset, of numbers. However, if we change certain things about the signal, would it necessarily change the label. For example:

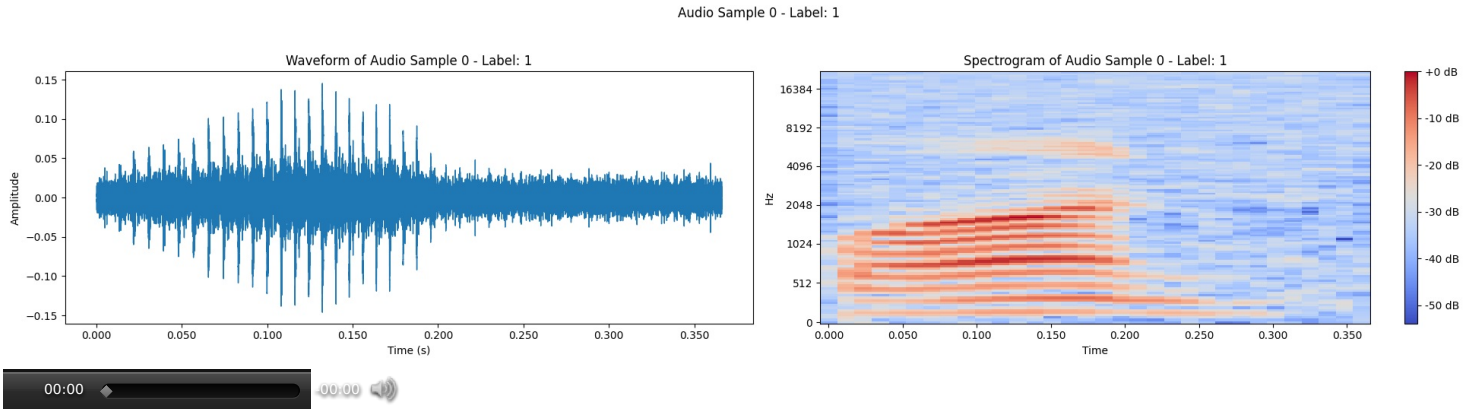
- speeding up or slowing down the speech still gives the same number
- shifting the pitch as well
- adding noise, making a worse recording should not change the outcome as well

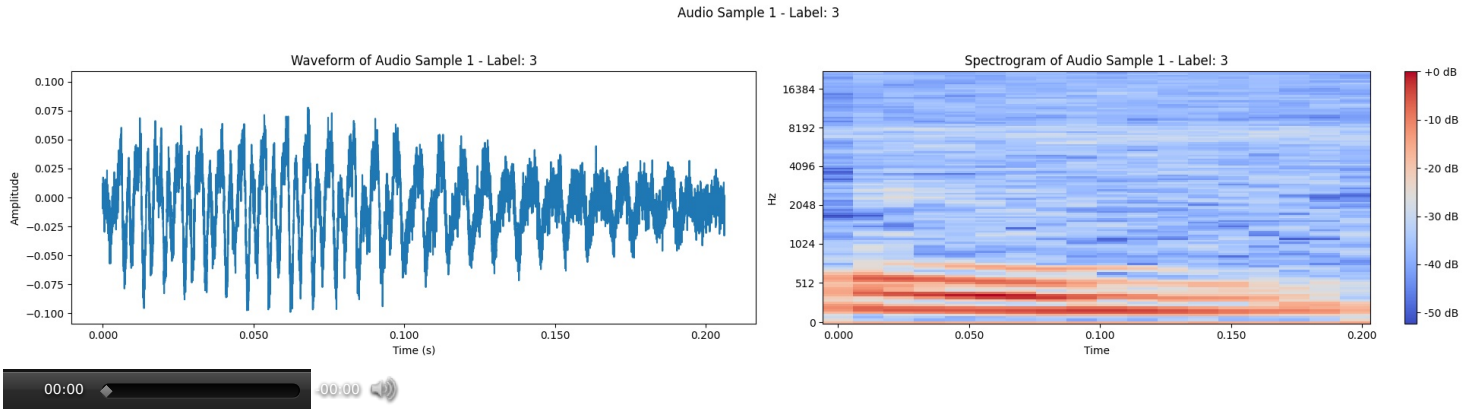
This is all a form of **data augmentation** - changes to our data that does not impact how our model should treat it, while being more adversarial and challenging for it. Below we have prepared one of the examples previously listed - adding Gaussian noise.

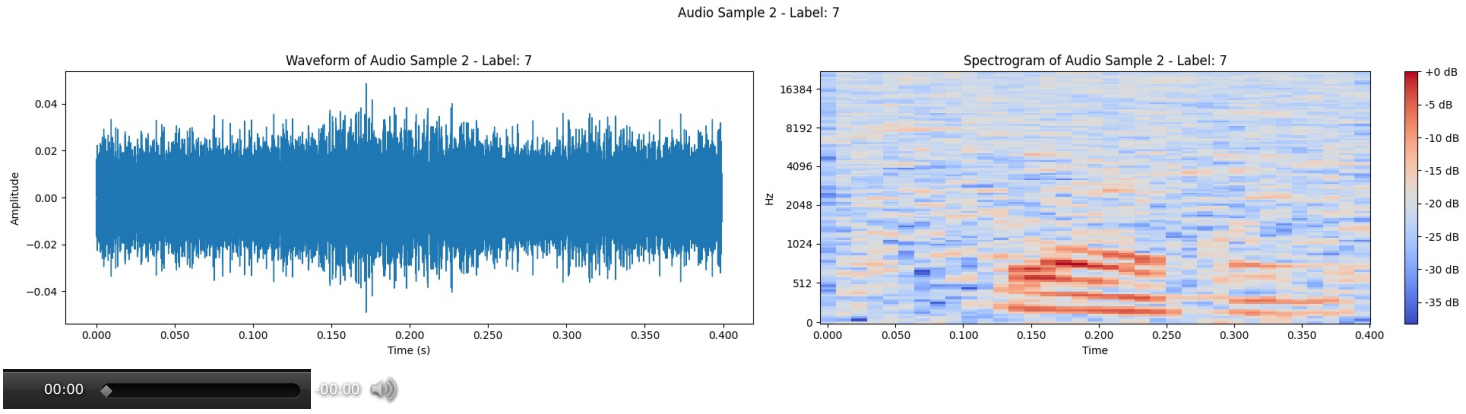
Below are two cells, the first one is a barebones augmentation function adding Gaussian noise to the signal, essentially perturbing it.

Take a listen and see if the numbers are still discernable. Also, try to examine how the signal looks visually different. Is there anything different about the wav file and the spectrograms.

```
100%|██████████| 1000/1000 [00:45<00:00, 22.00it/s]
```







Question:
How is the amplitude (the left plot) different for the augmented samples – why?
What would happen if we increase σ, which is currently $\sigma = 0.01$?

Answer:

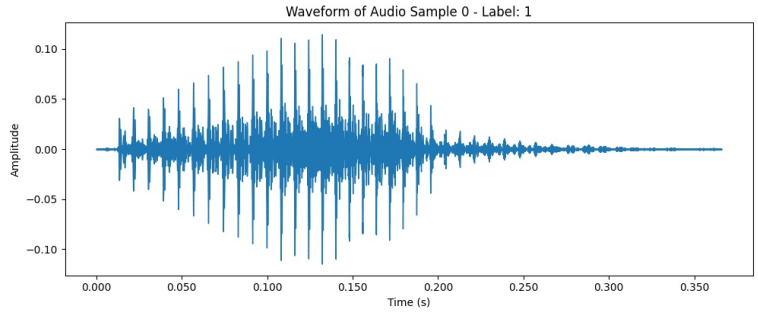
Due to the Gaussian noise, the amplitude randomly fluctuates up and down, and it sounds like a distortion. It is visible in the signal being more wavy.

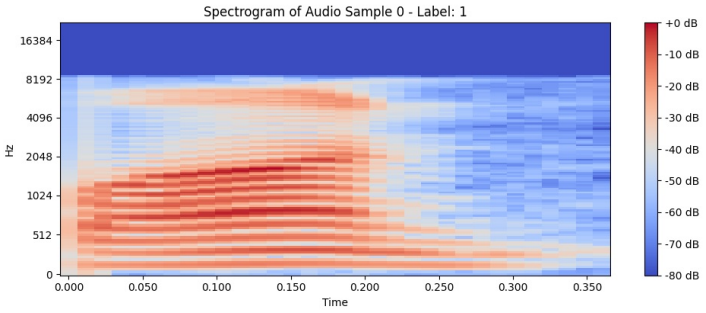
Additional Augmentations

Adding noise is not in this case the only transformation we can do to keep the label/nature of the audio the same. Below are some other examples. After them there are some questions about the transformations - explain why they work.

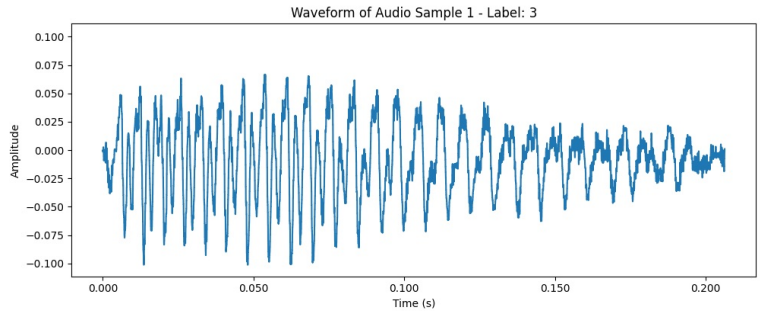
```
100%|██████████| 1000/1000 [00:30<00:00, 33.21it/s]
```

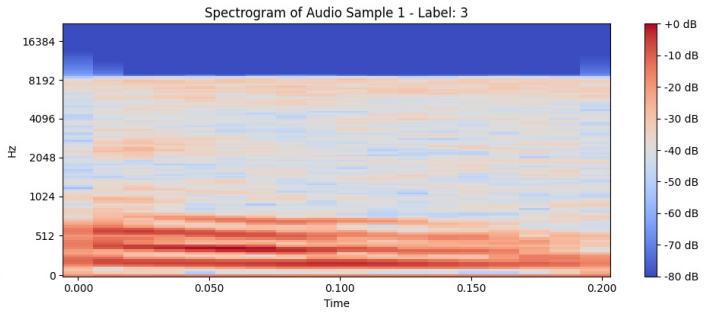
Audio Sample 0 - Label: 1



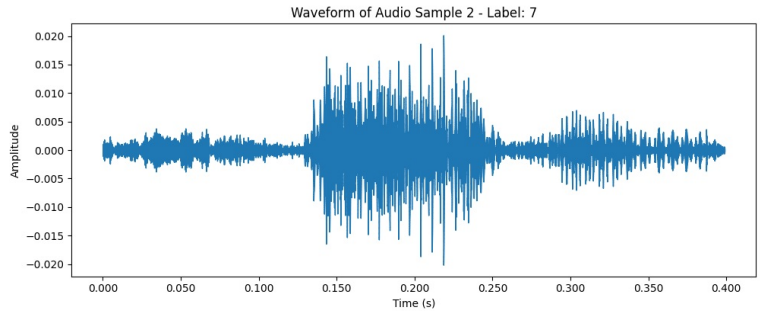


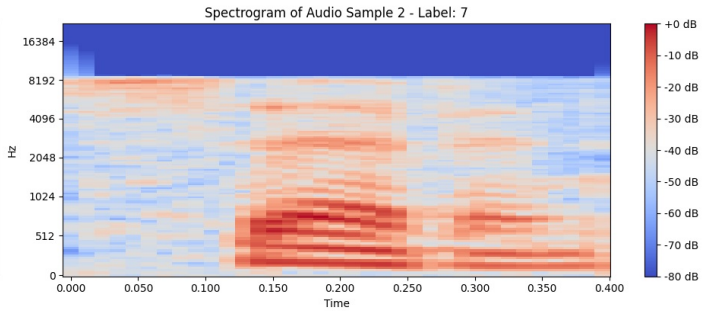
Audio Sample 1 - Label: 3





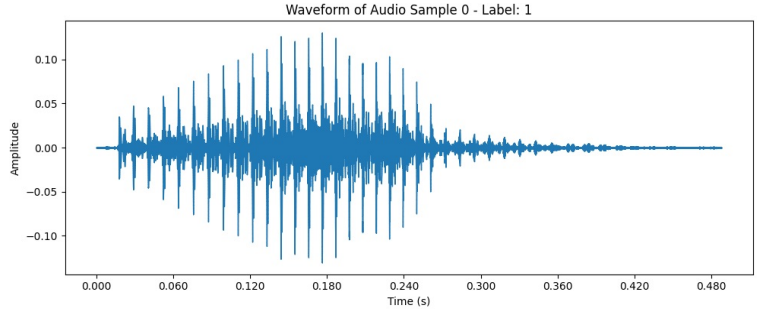
Audio Sample 2 - Label: 7

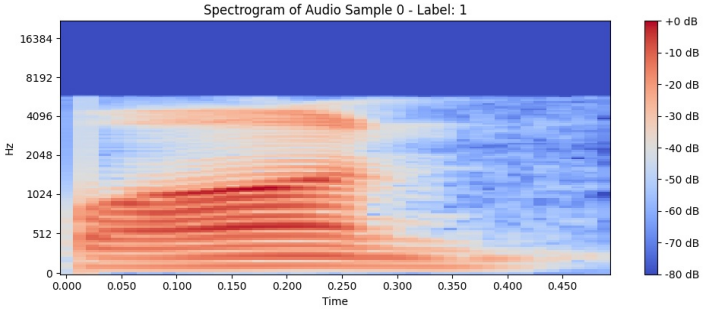


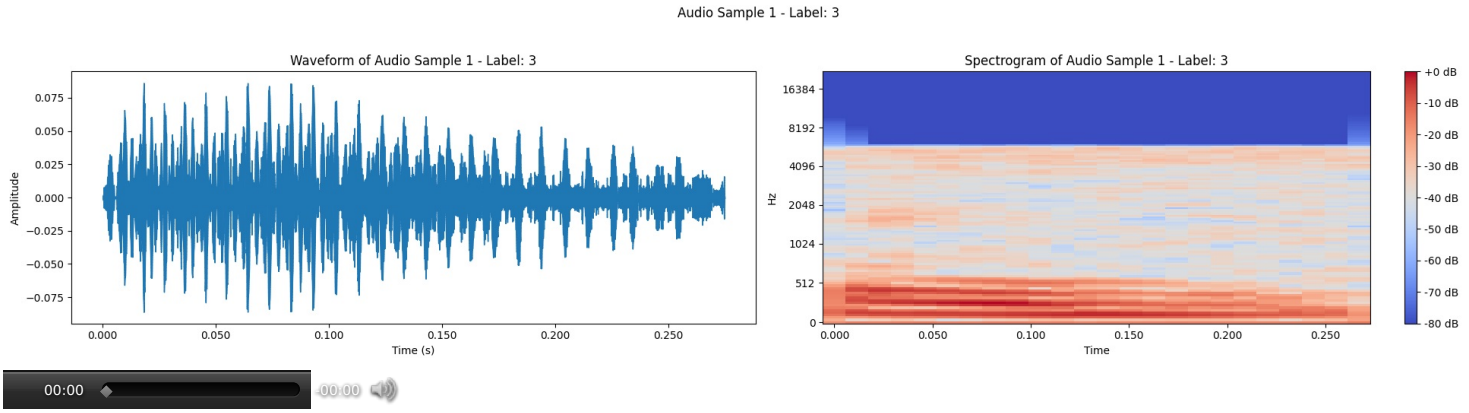


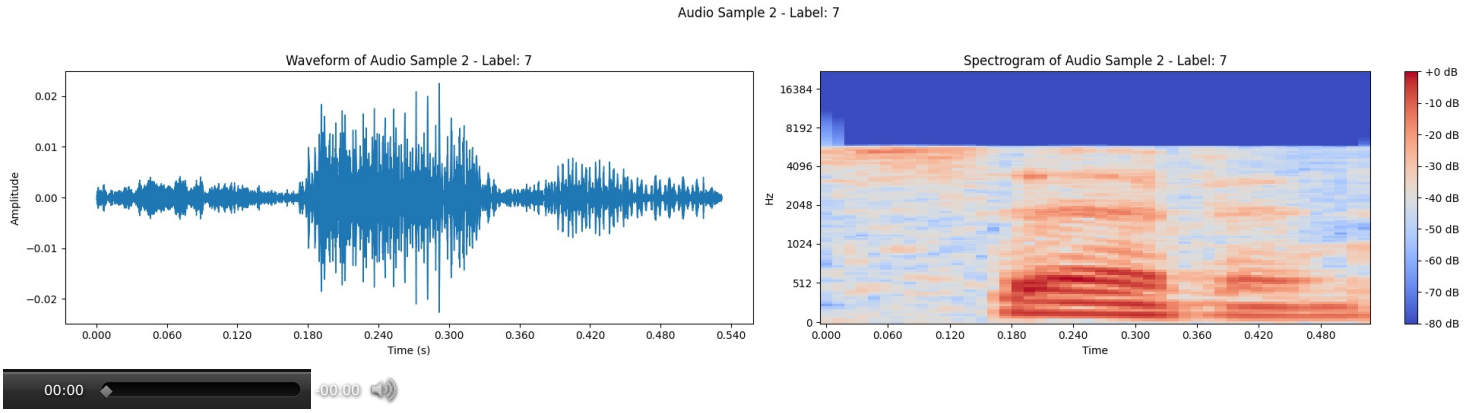
100%|██████████| 1000/1000 [00:21<00:00, 45.55it/s]

Audio Sample 0 - Label: 1









Question:
Explain what the additional two augmentations are in `augmentation_function_extra_1` and `augmentation_function_extra_2` are, and why they work.

Answer:

The amplitude regulates how high the pitch is at the current time step, while creating more interpreted values either speeds or slows down the playback.

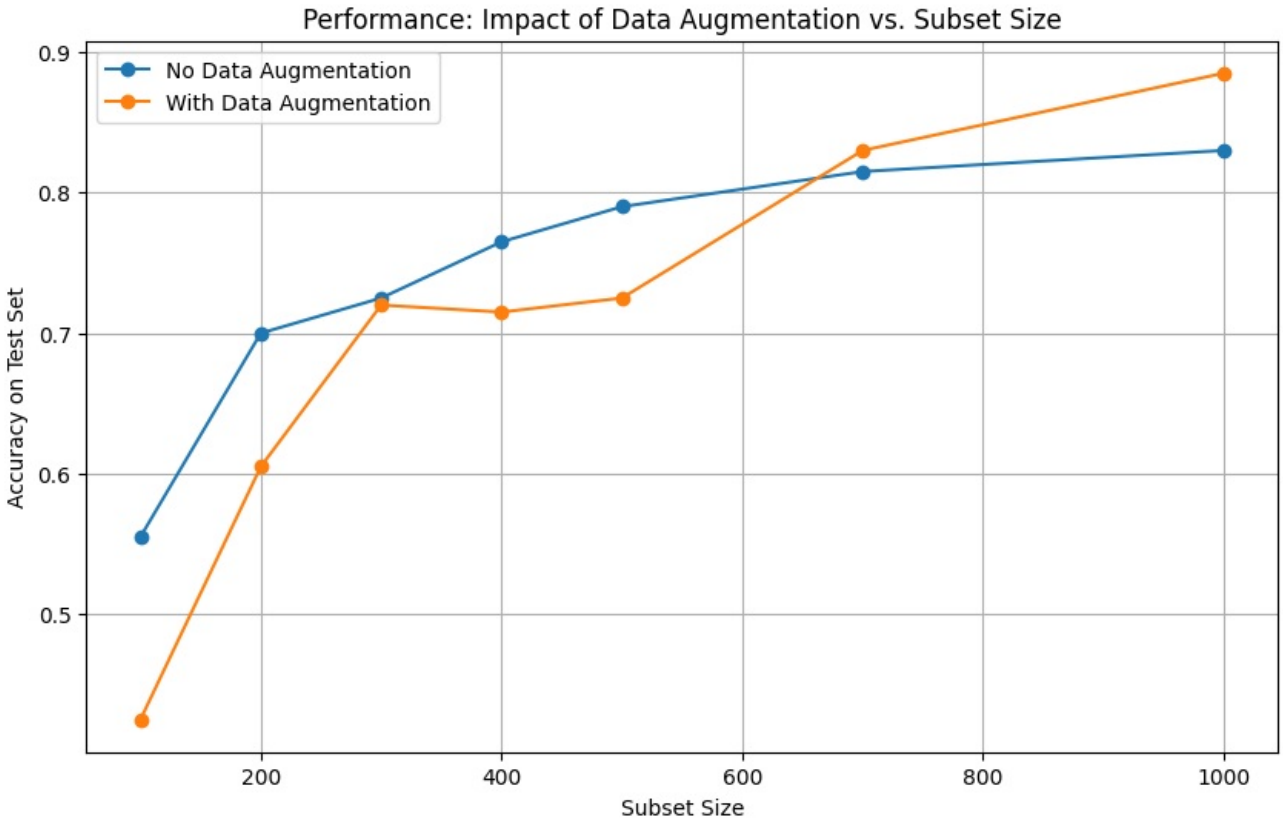
Below we have a final augmentation function using all three. Try to see how you can combine them all.

```
100%|██████████| 1000/1000 [00:27<00:00, 36.80it/s]
```

Training the Models

In this section, we give an illustrative example, of how training such a model works with and without data augmentations, and try to illustrate the *possible* benefits of it. Gradient Boosting Trees are used, but you do not need to concern yourself with their implementation for today.

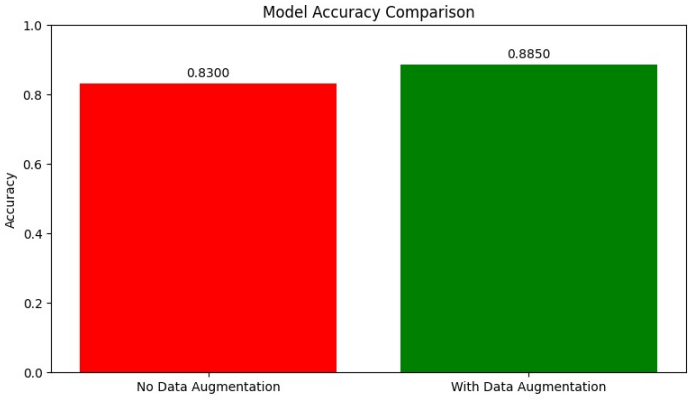
```
Accuracy without data augmentation: 0.8200
Accuracy with data augmentation: 0.8900
```

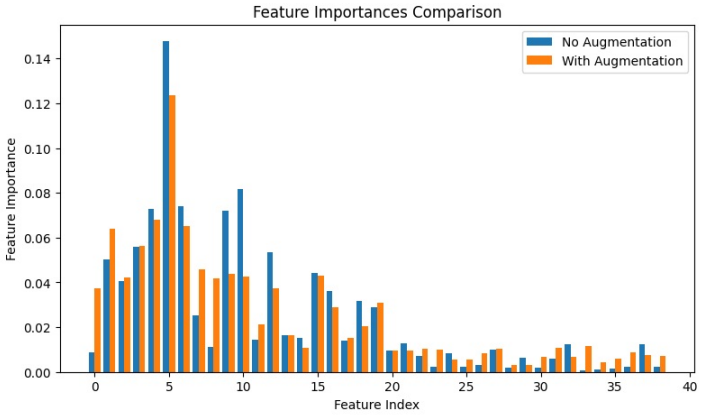


Question:
In the plot showing the performance of the models across different subset sizes (imitating different dataset sizes in general), please reason about why the two accuracies end up different from each other, even if we use data augmentation? Why is data augmentation lower in the beginning?

Answer:

It's only natural, the dataset size getting increased means we get a larger variety of samples, which ends up compensating for the missing data above. Data augmentation may be too much at times, and instead confuse the model when the data size is too small. At what point are we learning the noise and not the actual data. It is careful dance between augmenting too much and too little. However, we can see that after a certain threshold, it does end up making our model more robust on the same test set.





Theoretical Questions

Question:
In the plot showing the performance of the models across different subset sizes (imitating different dataset sizes in general), please reason about why the two accuracies end up different from each other, even if we use data augmentation? Why is data augmentation lower in the beginning?

Answer:

It's only natural, the dataset size getting increased means we get a larger variety of samples, which ends up compensating for the missing data above. Data augmentation may be too much at times, and instead confuse the model when the data size is too small. At what point are we learning the noise and not the actual data. It is careful dance between augmenting too much and too little. However, we can see that after a certain threshold, it does end up making our model more robust on the same test set.

Question:
Why are the feature importances for the augmented data more spread out than in the real data?

Answer:

The augmented data forces the model to look at more features, decreasing the amount of local patterns found in the data - thus it would try to examine the feature vector more thoroughly.

Question:
If you were to make a new data augmentation for audio, what would it be?

Answer:

Speedup, cropping, reverb, echo, etc. are all valid answers. In general, anything that wouldn't flip the label or make the data point unrecognizable is considered a good data augmentation.

Images Data Augmentation - MNIST

In this part of the notebook, we will show you a more popular example of data augmentation - namely on images and the MNIST Dataset. It is where data augmentation is more commonly applied in industry and more easily understood.

![]

Please run all code cells below, and you will end up with an interactive menu on the MNIST dataset. There you can play around with different data augmentation and have them visualized in real time. Then, run through the theoretical exercises and try to continue by improving the list of augmentations provided.

```
HBox(children=(VBox(children=(IntSlider(value=0, description='Rotation Angle [-180, 180]', layout=Layout(width…
```

Exercise: Creating Your Own Data Augmentation Pipeline

The time has come. After seeing our examples and playing around with how the given numbers look around, we want you to arrange your own set of augmentations. Your task is to find an augmentation to boost the performance on a training and validation set on a subset of data we offer you.

You will do all of this in the `transform_aug` variables, which takes in a list of augmentations. This list is currently empty (except for the normalization step, which we leave as an exercise for the reader to know why it is there). Throughout this notebook we have been using the *Albumentations* library to do our data augmentations. This will continue to be the case here. You are free to select from any of the following functions there, and adjust their parameters. Namely:

- `A.Rotate(float rotate, float p)`
- `A.Affine(float translate_x, float translate_y, float scale, float p)`
- `A.GaussianBlur(int gaussian_kernel_size, float blur_sigma, float p, bool gaussian_blur)`
- `A.HorizontalFlip(float p, bool horizontal_flip)`
- `A.VerticalFlip(float p, bool vertical_flip)`
- `A.RandomBrightnessContrast(float p, bool random_brightness_contrast)`
- `A.RGBShift(float p, bool rgb_shift)`

Important: Please take note that sometimes the function can take in both a number, a tuple, and a list!!

The API of the augmentations is provided on this link https://albumentations.ai/docs/api-reference/. Be careful - the optimal combination

may not be all of those augmentations at once, or with all of the parameters!

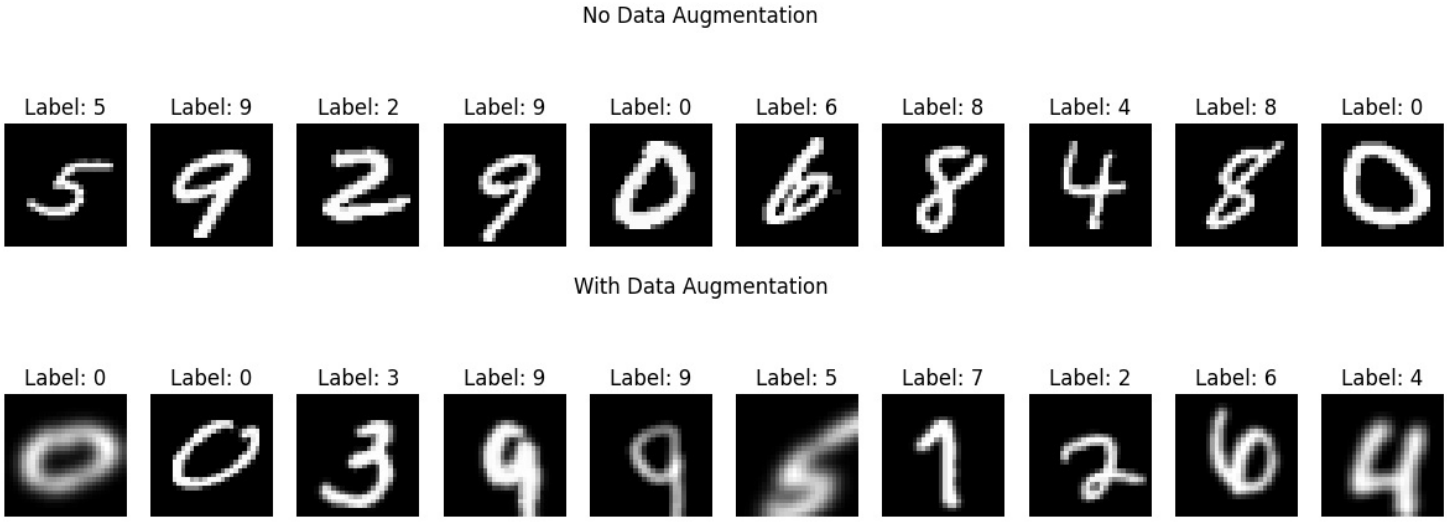
Question:
How often would you choose to apply the data augmentation with the parameter **p**? Justify. Think what happens if the values end up between 0 and 1.

Answer:

Except **p=0**, and arguably **p=1** there is no wrong answer. The more often you apply the data augmentation, the more you also lose the representation of your original data. In theory, that might be fine if the data augmentations are minor enough, but if excessive you would likely receive worse performance. If **p=0**, there's just no data augmentation, removing the point completely.

Dataset Parameters and Visualization

Here we set you the subset size you will be working on with your data augmentation from the code cell above. The expectation is that accuracy will increase for the correct set of data augmentation. Play around it - adjust the size of the dataset if you wish to or the samples per class, which attemt to balance out your distribution.



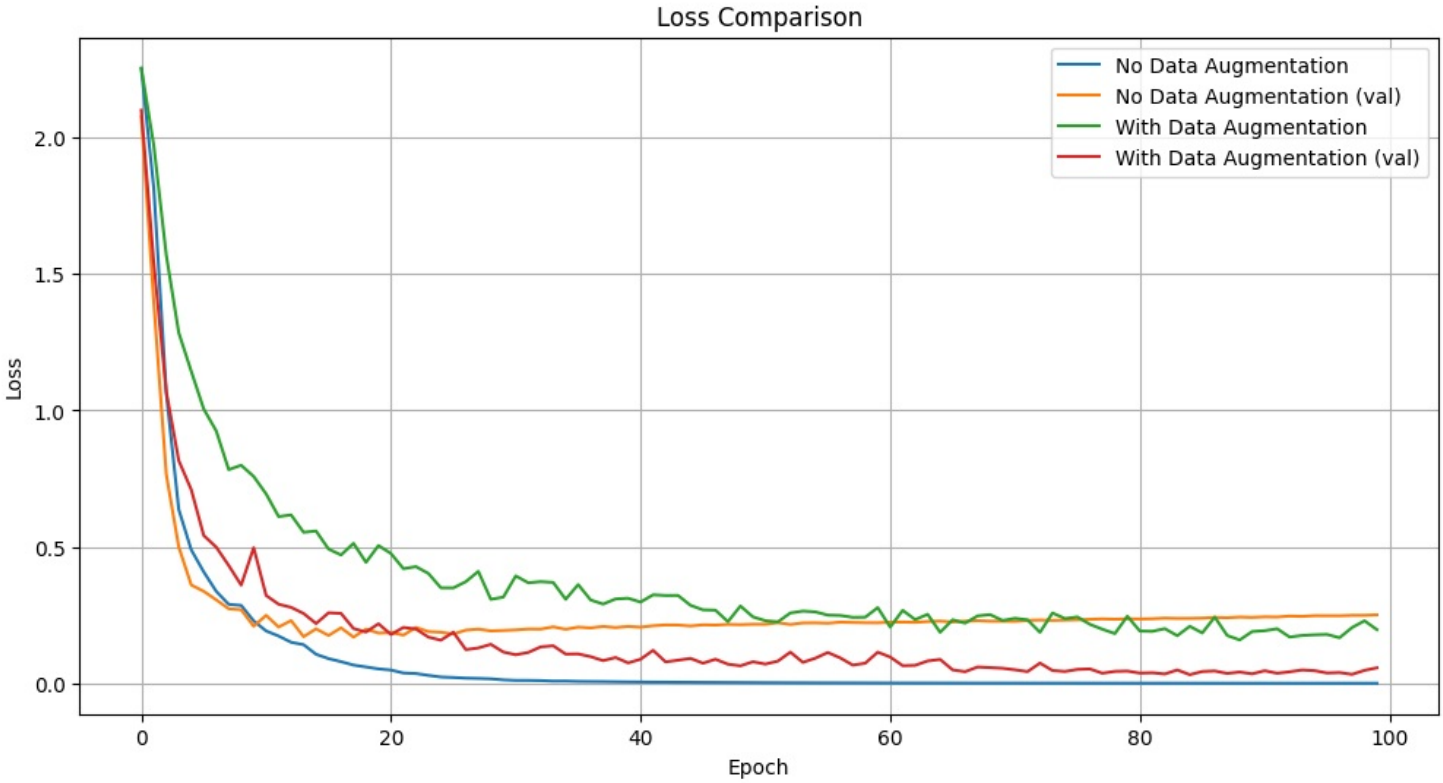

```
100%|██████████| 100/100 [00:32<00:00,  3.08it/s]
100%|██████████| 100/100 [01:00<00:00,  1.66it/s]
```

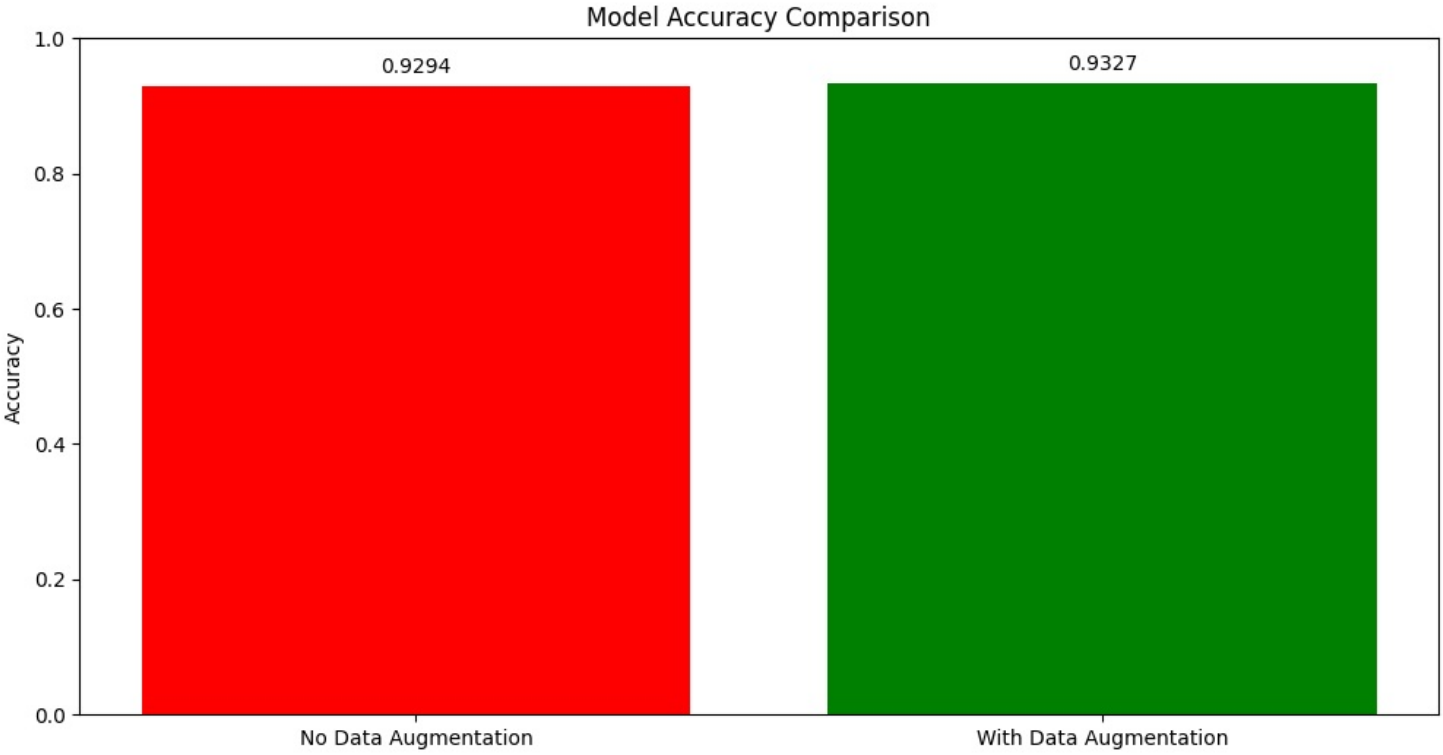
Question:
What data augmentations did you choose and why? Are there any data augmentations you did NOT choose? If so, why? Feel free to come back to this questions after experiemnting.

Answer:

We expect the students to use a combination of blurs, (moderate) rotations and affine transofmrations or anything to change the color distribution of the image. Performance should decrease in case of data augmentations that "flip" the label in some way (zooming in too much to an 8, roating a 6). Things like that confuse the models instead of making them more robust against noise.

```
Accuracy without augmentation: 0.9294
Accuracy with augmentation: 0.9327
```

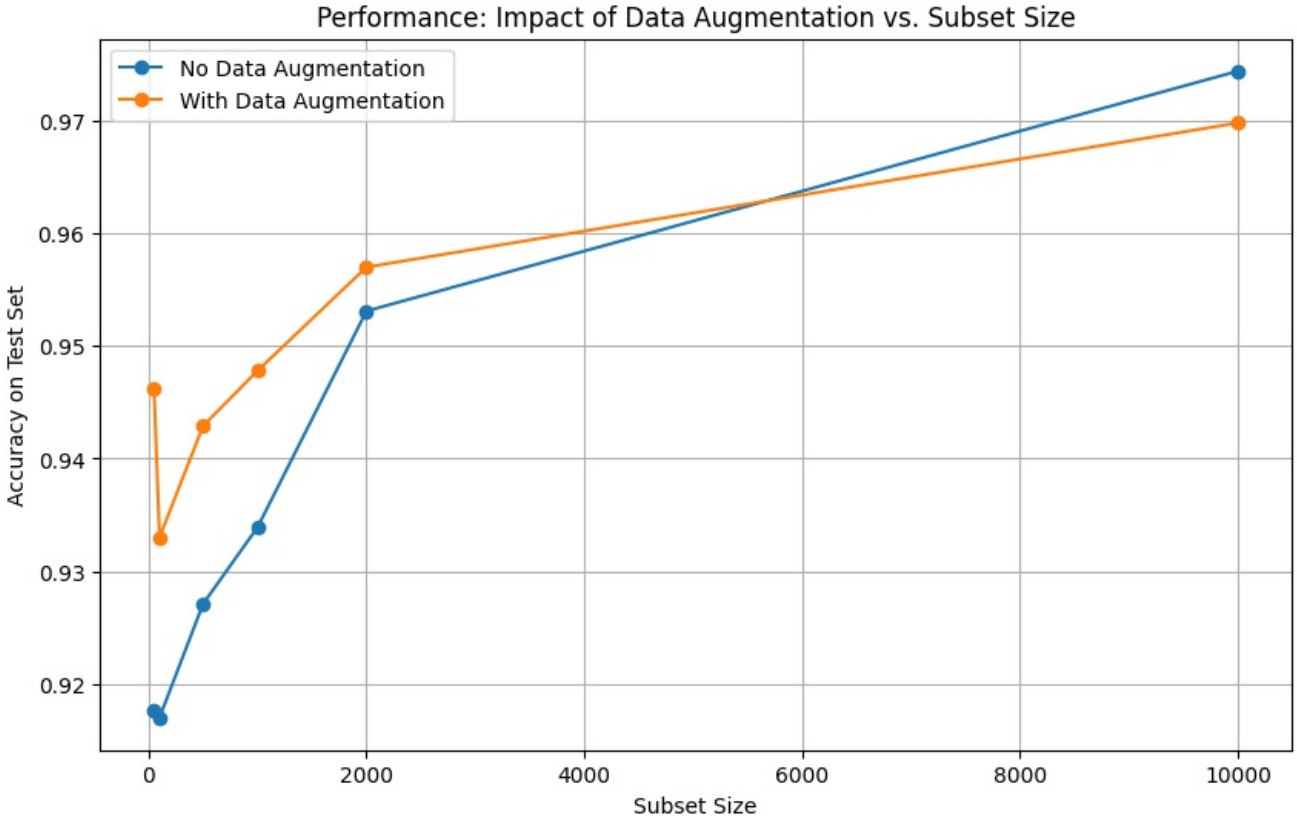




Question:
Examine the loss over different epochs (both validation and training) on the augmented and non-augmented data. What difference do you see? Why do you see it?

Answer:
The augmented loss is much less stable than the non-augmented one. This makes sense as augmentations change over forward passes and the task is overall mcuh harder.

```
100%|██████████| 10/10 [00:00<00:00, 52.16it/s]
100%|██████████| 10/10 [00:00<00:00, 27.29it/s]
100%|██████████| 10/10 [00:00<00:00, 28.08it/s]
100%|██████████| 10/10 [00:00<00:00, 13.18it/s]
100%|██████████| 10/10 [00:01<00:00,  6.09it/s]
100%|██████████| 10/10 [00:03<00:00,  3.12it/s]
100%|██████████| 10/10 [00:02<00:00,  3.34it/s]
100%|██████████| 10/10 [00:06<00:00,  1.53it/s]
100%|██████████| 10/10 [00:06<00:00,  1.61it/s]
100%|██████████| 10/10 [00:13<00:00,  1.31s/it]
100%|██████████| 10/10 [00:33<00:00,  3.36s/it]
100%|██████████| 10/10 [01:00<00:00,  6.01s/it]
```



Question:
Reason about the performance on the augmented data over the different subset sizes on the plot. When is no augmentation better?

Answer:
There are points when the amount of data shown is sufficient for the task at hand. Then the data augmentation is most likely distancing the model more from reality than is fully necessary. The best thing to do for your model, of course is to add more data - data augmentation is a remedy, but not a replacement.

Contrail Application

Now comes the time to apply our data augmentations to real data, not just a toy problem like MNIST.

We are in luck, however! We can (mostly) just reapply all of the data augmentations we have done to MNIST. The following section details how data augmentation is applied and lets you play with another set of augmentations to improve contrail performance.

Alternatively, you could also take a look at our set of data augmentatiosn they are not necessarily the best set of augmentations for all cases. Please answer all of the theoretical questions, and run through the code. Our configuration of data augmentations is not necessarily optimal.

```
HBox(children=(VBox(children=(IntSlider(value=0, description='Rotation Angle [-180, 180]', layout=Layout(width…
```

Intermezzo: Transfer Learning

Transfer learning is a machine learning technique where knowledge learned from a previous task is applied to a new one. Intuitively, if these tasks are related, parts can be re-used between them. In the context of machine and deep learning (specifically with neural networks), this very often means using or starting off with previously learned weights on a similar task. An example task could be a classifier on cars that

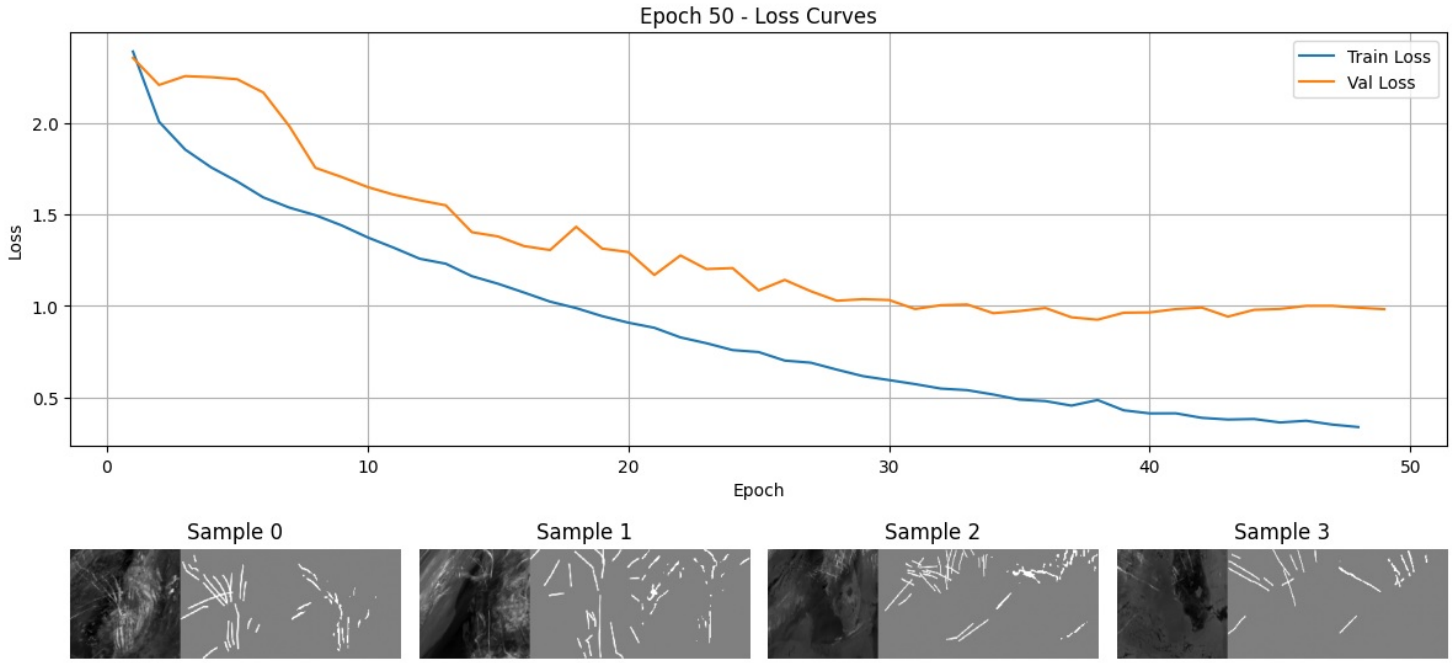
you now wanna apply on trucks.

This is also what happens in the paper we discuss in these exercises. It uses what they call "few-shot learning" for contrail detection where the initial weights of the model used here are from segmentation on ImageNet, one of the largest image datasets and most famous benchmarks in all of deep learning. It is a "starting point" to whatever we end up doing, as long as the tasks are related.

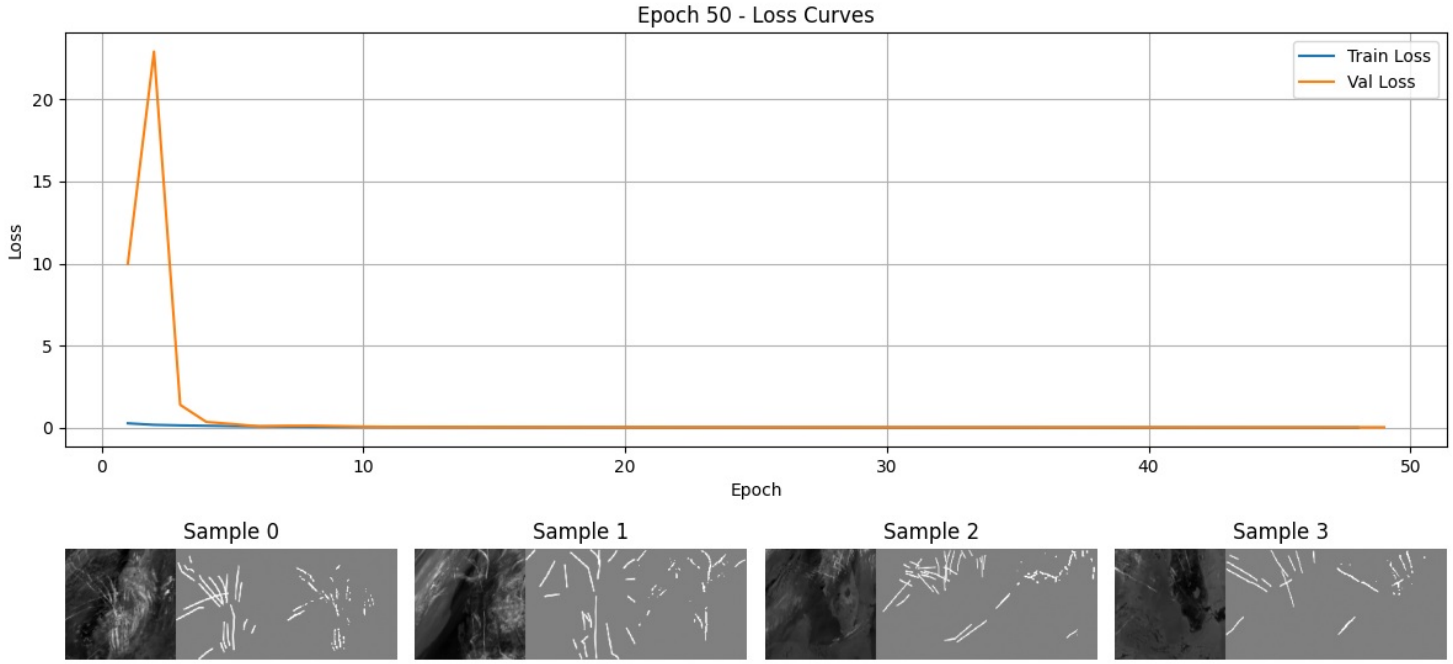
We mention it here as **it is related to transfer learning** - it once again is used in cases with little data and helps reduce generalization error by including example from multiple datasets.

Some resources for the curious ones are listed below:

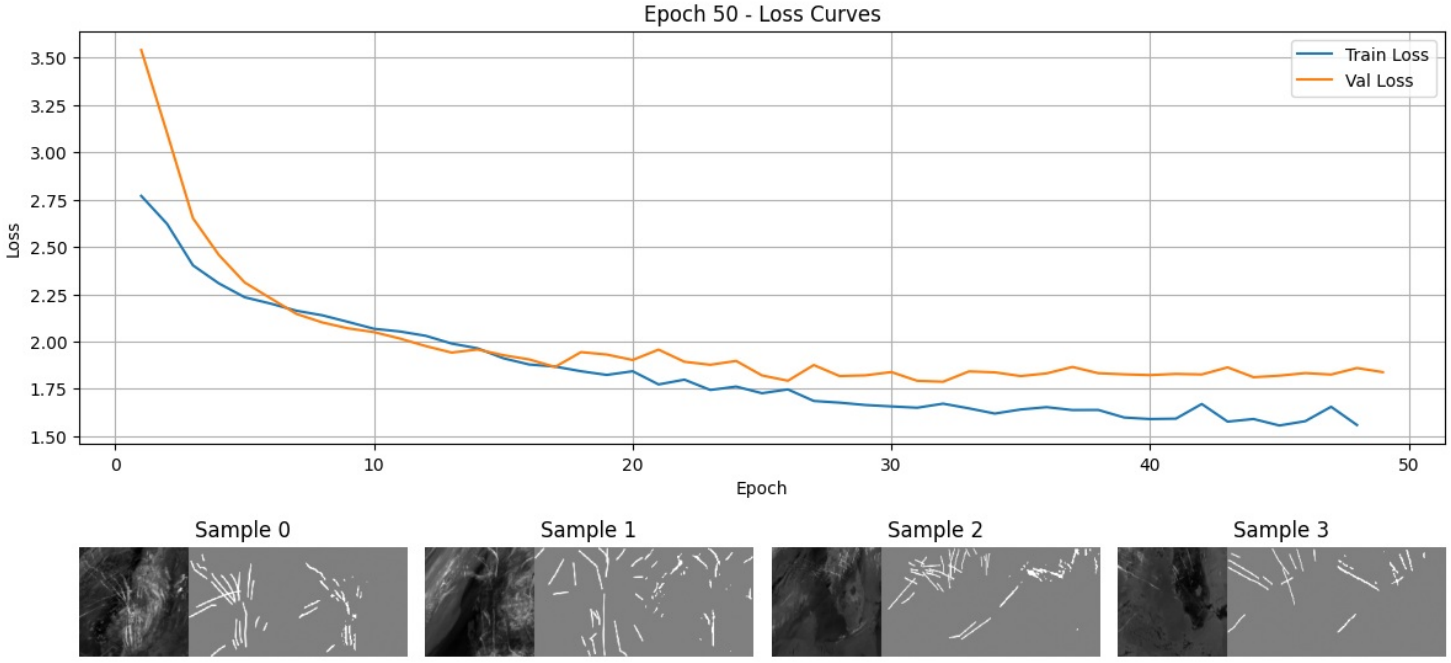
- https://www.ibm.com/think/topics/transfer-learning
- https://medium.com/@davidfagb/guide-to-transfer-learning-in-deep-learning-1f685db1fc94
- https://arxiv.org/abs/2104.02144



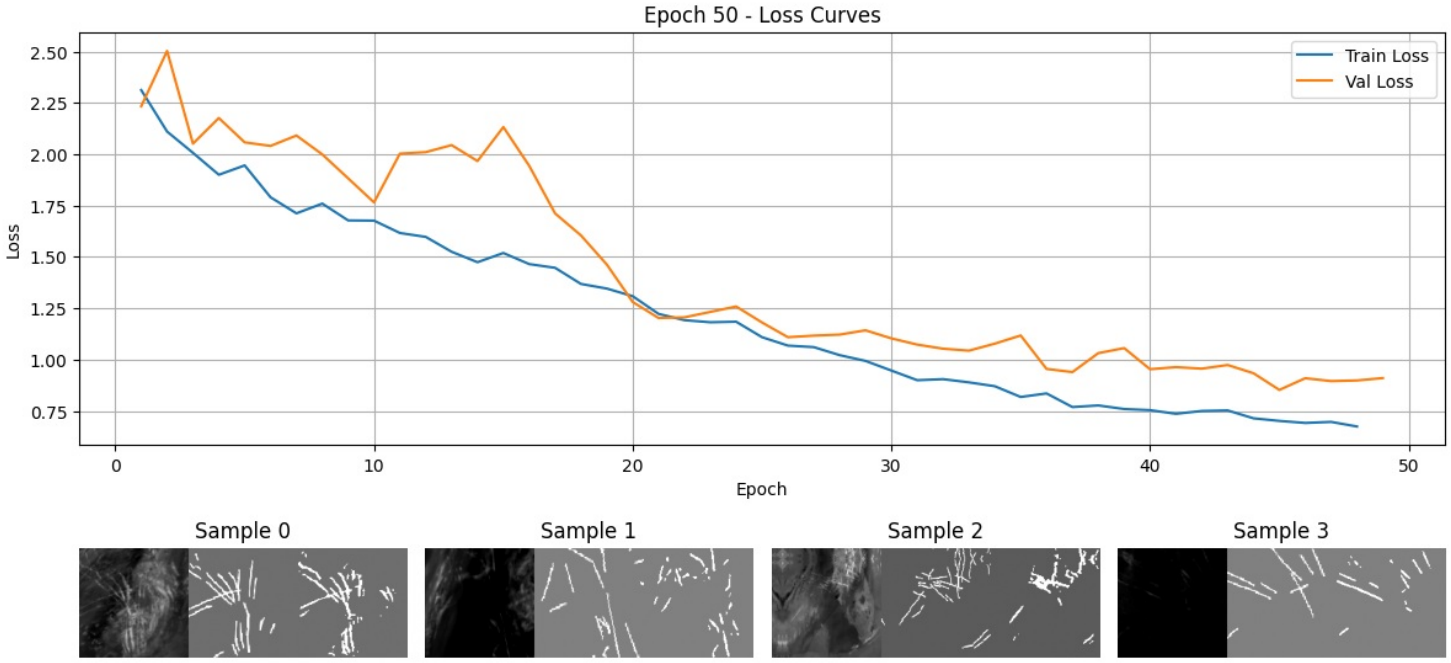

```
`Trainer.fit` stopped: `max_epochs=50` reached.
```



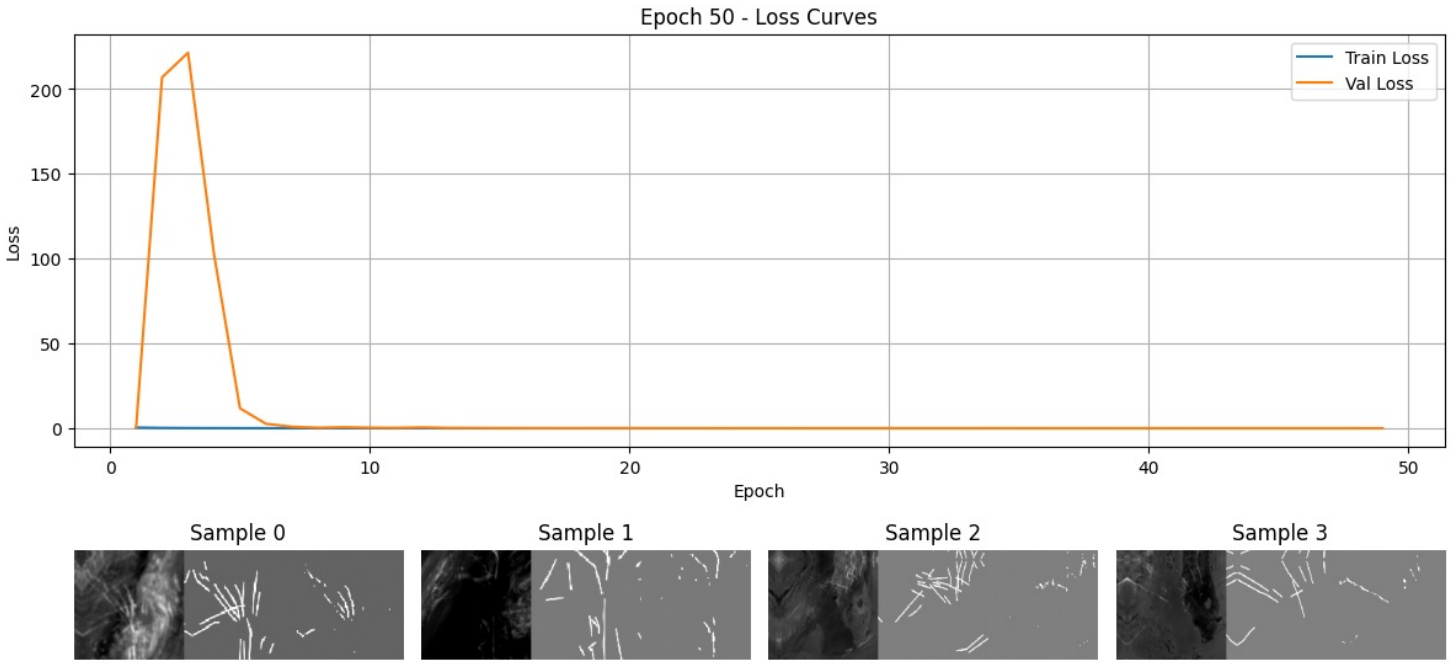

```
`Trainer.fit` stopped: `max_epochs=50` reached.
```



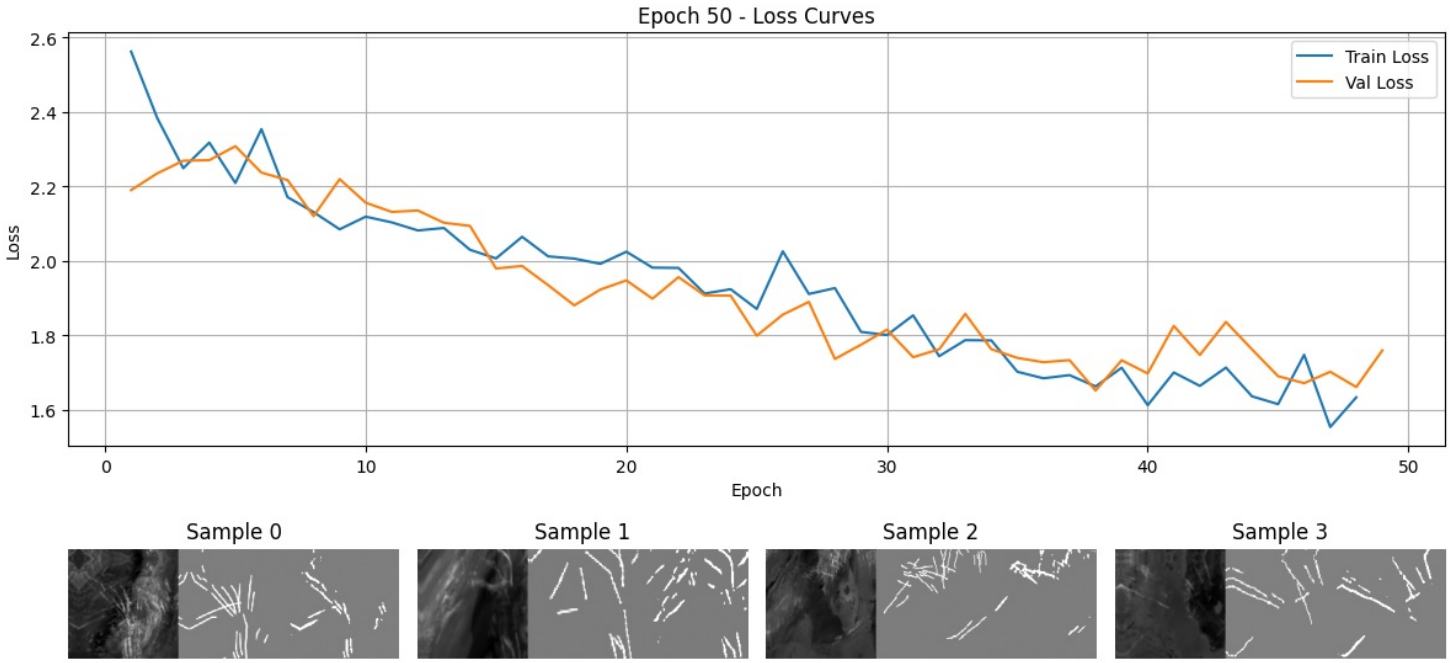
`Trainer.fit` stopped: `max_epochs=50` reached.



`Trainer.fit` stopped: `max_epochs=50` reached.



```
`Trainer.fit` stopped: `max_epochs=50` reached.
```




```
`Trainer.fit` stopped: `max_epochs=50` reached.
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 1
----> 1 from evaluate import evaluate_contrail_model
      2 a = A.Compose([
      3             A.Resize(128, 128),
      4             A.Normalize(mean=0.5, std=0.5, max_pixel_value=255.0),
      5         ])
      7 evaluate_contrail_model("data/models/dice:base-100epoch.torch", dataset="own", augmentation=a, num_images=4)

ModuleNotFoundError: No module named 'evaluate'
```