oduct is in a pre-release state and might change or have limited support. For more information, see the product laun
s (/products/#product-launch-stages).

This page describes best practices around preparing your data, evaluating your models, and improving model performance.

- The data that you use for training should be as close as possible to the data that you want to make predictions upon. For example, if your use case involves blurry and low-resolution videos (such as security camera footage), your training data should include blurry, low-resolution videos. In general, you should also consider providing multiple angles, resolutions, and backgrounds for your training videos.
- Your training data should meet certain minimal requirements:
    - All of your data should have labels.
    - Your labels must be valid strings (no commas).
    - All video segments should have valid timestamps. Your segments must have a start and an end, the end time must be greater than 0 and less than the total duration of the video, and the start must be less than the end time.
    - All video URIs in your CSV must be stored in an accessible Cloud Storage bucket.
    - Your data should include at least 2 different classes, with each class having at least 10 items.
- The more training and test data that you have, the better. The more powerful the model is, the more data hungry it becomes.
- The amount of data needed to train a good model depends on different factors:
    - The amount of classes. The more unique classes you have, the more samples per class are needed.

    - Complexity or diversity of classes. Neural networks might quickly distinguish between videos of running and swimmming, but would need a lot more samples to classify 30 different dancing styles.
    - Multi-label models are more difficult to train than multi-class models. If most of your cases have only 1 label per sample, consider using a multi-class model training instead of multi-label.
- As a rule of thumb, you should have at least 200 training samples per class if you have distinctive and few classes, and more than 1000 training samples if the classes are more nuanced and you have more than 50 different classes.
- Avoid training a model with highly imbalanced data. In many cases, the number of samples per class is not equal. While the differences are not big, it's not that bad. However, when there is a bigger imbalance—for example, some classes present more than 10 times more often than others—this becomes a problem. While AutoML Video Classification attempts to correct for class imbalances, it's not an ideal configuration for model training.

To learn more, see the information on preparing your data (/video-intelligence/automl/docs/prepare).

In machine learning, you usually divide your datasets into three separate subsets: a training dataset, a validation dataset, and a test dataset. A training dataset is used to build a model. The model tries multiple algorithms and parameters while searching for patterns in the training data. As the model identifies patterns, it uses the validation dataset to test the algorithms and patterns. The best performing algorithms and patterns are chosen from those identified during the training stage.

After the best performing algorithms and patterns have been identified, they are tested for error rate, quality, and accuracy using the test dataset. You should have a separate test dataset that you can use to test your model independently.

Both a validation and a test dataset are used in order to avoid bias in the model. During the validation stage, optimal model parameters are used, which can result in biased metrics. Using the test dataset to assess the quality of the model after the validation stage provides an unbiased assessment of the quality of the model.

Use the following best practices when splitting your data:

- We recommend that all datasets (also called dataset splits) represent the same population, have similar videos, with similar distribution of labels.

When you provide your data, AutoML Video Classification can automatically split it into training, validation, and testing datasets. You can assign the train split labels yourself as well. Either all samples should come from a single video, or from many videos. Having only a few videos could be dangerous if the videos are different or if the label distribution is not the same within the videos. This is because when AutoML Video Classification is generating the train-validation-test splits, it does this on video level.

For example, if you have only 3 videos with thousands of annotated video segments inside, but some classes are present only in individual videos, it can happen that the model doesn't train for some labels and thus can miss those labels during prediction.

- Avoid data leakage. Data leakage happens when the algorithm is able to use information during model training that it should not and that information will not be available during future predictions. This can lead to overly optimistic results on train, validation, and test datasets; but might not perform as well when asked to make predictions future unseen data.

  Some leakage examples include: a bias based upon camera viewing angle or light conditions (morning/evening); a bias towards videos that have commentators versus those that don't; a bias based upon videos with certain labels that come from specific regions, language groups, commentators, or that include the same logo.

  To avoid data leakage, do the following:

  - Have a well-diversified set of videos and video segment samples.
  - Review the videos to make sure there are no hidden hints (for example, videos with positive samples were taken during the afternoon, while videos with negative samples were taken during the morning).

To learn more, see the information on preparing your data (/video-intelligence/automl/docs/prepare).

For an example of data source, see the following publicly available video datasets:

- UCF-101 (http://crcv.ucf.edu/data/UCF101.php): Action Recognition Data Set (101 classes; multi-class; 13320 short videos/samples)
- HMDB-51 (http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/): Human Motion DataBase (51 classes; multi-class; 6849 short videos/samples)
- Kinetics (https://deepmind.com/research/open-source/open-source-datasets/kinetics/): A large scale dataset for action recognition (400/600/800 classes; multi-class; 400000+ short videos/samples)

- Something-Something (https://20bn.com/datasets/something-something): Human actions with objects (174 classes; 220847 videos/samples)
- AVA (https://research.google.com/ava/): densely annotated Atomic Visual Actions (80 classes; multi-label; 1.58M annotations within 430 15min video clips)

The same data can be used to train different models and to generate different prediction types, depending on what you need. As well, training the same model on the same data might lead to slightly different results. The neural network model training involves randomized operations, so that one cannot guarantee to train exactly the same model with the same inputs, and the predictions might be slightly different.

To learn more, see the information on managing models (/video-intelligence/automl/docs/models).

Once your model finished training, you can evaluate the performance on validation and test datasets or on your own new datasets.

Common classification metrics include:

- Area Under the Curve (AUC), also known as the "Area under the curve of the Receiver Operating Characteristic." The curve plots the recall over the probability of false alarm at different score thresholds. The range of AUC values is between 0.5 and 1.0. This metric is mostly used for binary classification problems.
- Area Under the Precision/Recall Curve (AuPRC), also known as average precision (AP). It is the integral of precision values over the range of the recall values. It is best interpreted for binary problems.
- Mean average precision (mAP or MAP) can be considered as the mean of average precision (AP) metrics over multiple classes or labels. Sometimes, mAP and AP are used interchangeably.
- Accuracy represents the ratio of samples that are correctly classified.
- For binary and multi-class problems, you can also examine precision and recall at various confidence score thresholds independently.

- If there are not too many multi-class labels, you can examine the confusion matrix which shows which labels were mis-classified to which predictions. Alternatively the UI can show you the top misclassified classes within the confusion matrix.

Not all metrics can be used for the different video classification problems. For example, the intuitive understanding of precision and recall from a binary classification becomes more ambiguous when considering multiple classes (multi-class problem) or if there are multiple valid labels per sample (multi-label problem). Especially for the later case there is no single and well adopted metric. However, you might consider the Average Precision metric for evaluation decisions.

However, keep in mind that it's overly simplified to evaluate model performance based on a single number or metric. Consider looking at different metrics and as well at the plots of precision-recall curves, for example, or the distributions of misclassified samples such as in the confusion matrix.

Here are some other helpful tips when evaluating your model:

- Be mindful of the distribution of labels in your training and testing datasets. If the datasets are imbalanced, high accuracy metrics might be misleading. Since by default every sample has the same weight during evaluation, a more frequent label can have more weight. For example, if there are 10 times more positive labels then negatives, and the network decides to simply assign all samples to the positive labels, then you can still achieve an accuracy of 91%, but it doesn't mean that the trained model is that useful.
- You can also try to analyze ground truth and prediction labels for example in a Python script using scikit-learn (https://scikit-learn.org/stable/). There, you could look into different ways to weight the labels during evaluation: common approaches include macro averaging (metrics are computed per class, and then averaged), weighted (metrics are computed per class, and then averaged based with weights based on the frequency of individual classes), micro (each sample has the same weight, independent of any potential imbalance).
- Debugging a model is more about debugging the data than the model itself. If at any point your model starts acting in an unexpected manner as you're evaluating its performance before and after pushing to production, you should return and check your data to see where it might be improved.

To learn more, see the information on evaluating your model (/video-intelligence/automl/docs/evaluate).

On the **Evaluate** tab in the AutoML Video Classification UI (https://console.cloud.google.com/video-intelligence/datasets), you can assess your custom model's

performance using the model's output on test examples, and common machine learning metrics. The tab displays the following information about your model:

- The model output
- The score threshold
- True positives, true negatives, false positives, and false negatives
- Precision and recall
- Precision/recall curves.
- Average precision

When reading the model evaluation data in the Google Cloud Console, keep the following in mind:

- AutoML Video Classification pulls example video segments from your test data to present entirely new challenges for your model. For each example, the model outputs a series of numbers that communicate how strongly it associates each label with that example. If the number is high, the model has high confidence that the label should be applied to that document.
- You can convert the confidence numbers into a binary 'on/off' value by setting a *score threshold*. The score threshold refers to the level of confidence the model must have to assign a category to a test item. The score threshold slider in the UI is a visual tool to test the impact of different thresholds for all categories and individual categories in your dataset. If your score threshold is low, your model classifies more video segments, but runs the risk of misclassifying a few video segments in the process (fewer false negatives, but many false positives). If your score threshold is high, your model classifies fewer video segments, but it has a lower risk of misclassifying video segments (few false positives, but many false negatives). You can tweak the per-category thresholds in the UI to experiment. However, when using your model in production, you must enforce the thresholds you found optimal on your side.
- After applying the score threshold, the predictions made by your model fall in one of the following four categories:
    - True positive: The model predicted the label correctly.
    - True negative: The model correctly didn't predict this label.
    - False positive: The model wrongly predicted the label for a video segment.
    - False negative: The model failed to predict the true label.

    You can use these categories to calculate precision and recall—the metrics that help you gauge the effectiveness of your model.

- Precision and recall help you understand how well your model is capturing information, and how much it's leaving out. The precision score measures, from all the test examples that were assigned a label, how many actually were supposed to be categorized with that label. The recall score measures, from all the test examples that should have had the label assigned, how many were actually assigned the label.
- You can compare the model's performance on each label using a confusion matrix. In an ideal model, all the values on the diagonal are high, and all the other values are low. This shows that the desired categories are being identified correctly. If any other values are high, it indicates how the model is misclassifying test images.
- The score threshold tool allows you to explore how your chosen score threshold affects your precision and recall. As you drag the slider on the score threshold bar, you can see where that threshold places you on the precision-recall tradeoff curve, as well as how that threshold affects your precision and recall individually. For multiclass models, on these graphs, precision and recall means the only label used to calculate precision and recall metrics is the top-scored label in the set of labels returned. This can help you find a good balance between false positives and false negatives.

  Once you've chosen a threshold that seems to be acceptable for your model on the whole, you can click individual labels and see where that threshold falls on their per-label precision-recall curve. In some cases, it might mean you get a lot of incorrect predictions for a few labels, which might help you decide to choose a per-class threshold that's customized to those labels.
- A useful metric for model accuracy is the area under the precision-recall curve. It measures how well your model performs across all score thresholds. In AutoML Video, this metric is called Average Precision. The closer to 1.0 this score is, the better your model is performing on the test set; a model guessing at random for each label would get an average precision around 0.5.

AutoML Video Classification uses 30% of your data automatically—or, if you chose your data split yourself, whatever percentage you opted to use—to test the model. The **Evaluate** tab in the AutoML Video Classification UI (https://console.cloud.google.com/video-intelligence/datasets) tells you how the model did on that test data. But just in case you want to sanity-check your model, there are a few ways to do it. One way is to provide a CSV file with video data for testing in the "Test & Use" tab, and look at the labels the model predicts for the videos. Hopefully, this matches your expectations.

You can adjust the threshold for the predictions visualization and as well to look the predictions at 3 temporal scales: 1 second intervals, video camera shots after automated shot boundary detection, and entire video segments.

If you get your initial model performance and would want to continue to improve it, you can try a few different approaches:

- Increase the number of labeled samples (especially for under-represented classes).
- Closely examine where your model performs not so well:
    - Maybe a class is too broad and it would make sense to split it into two or more classes?
    - Or maybe some classes are too specific and could be merged without affecting the final goal of the project?
    - Consider labeling more samples particularly for the classes which are performing worse.
- If applicable, switch from multi-class to multi-label problem, since usually it is easier to train a neural network model that tries to classify samples into disjoint classes than to try to predict for each sample a set of labels of unknown length.
- Reduce data imbalance. Either add more samples or potentially try to reduce the number of samples of high frequency class, particularly in cases when there is a big imbalance, for example, 1-to-100 or more.
- Check carefully and try to avoid any potential data leakage.
- Drop less important classes to concentrate on fewer critical ones.
- Review other options available to you on the Support (/video-intelligence/automl/docs/support) page.