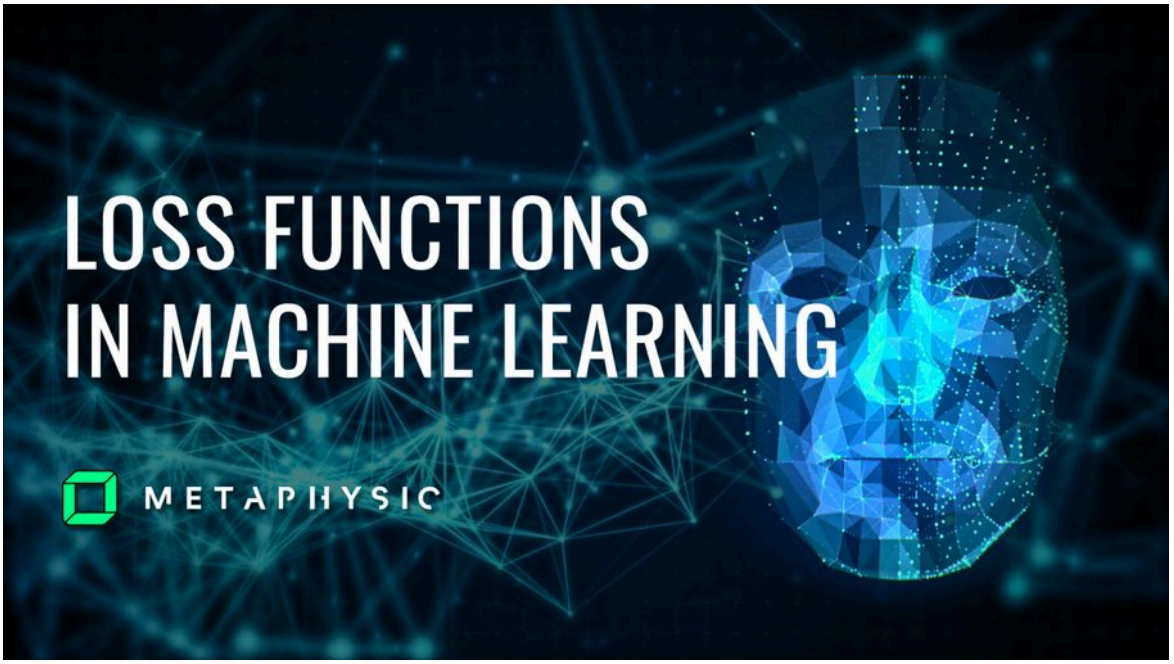
Loss Functions in Machine Learning

Originally published February 3, 2023 at https://blog.metaphysic.ai/loss-functions-in-machine-learning/

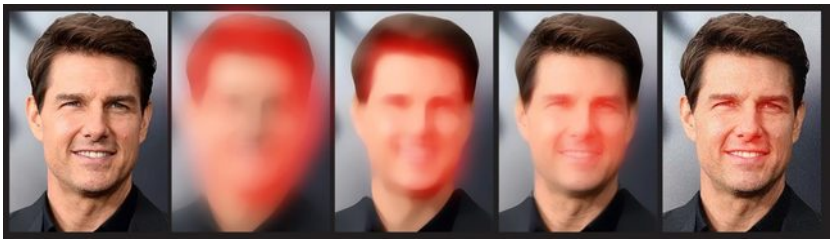


In machine learning, *loss functions* (also known as *cost functions*) are used during the training of a new AI model to determine how the model is doing at learning to make good predictions about the data it's being trained on.

Loss functions make guesses about the data that's being trained, known as *predictions*; the more advanced the training, the more information the loss function will have, in order to make better predictions.

One example, close to our own area of interest, can be found in the training of a deepfake model. In this scenario, the autoencoder network is being asked to gain a broad understanding of a single identity, and to learn how to recreate it (it's only after the model is trained that it is asked to 'switch' the trained identity with another one).

The system does this in much the same way as an artist, initially laying down general swathes of shape and color, and gradually intensifying the level of detail until the internal neural representation is (hopefully) faithful to the source identity:

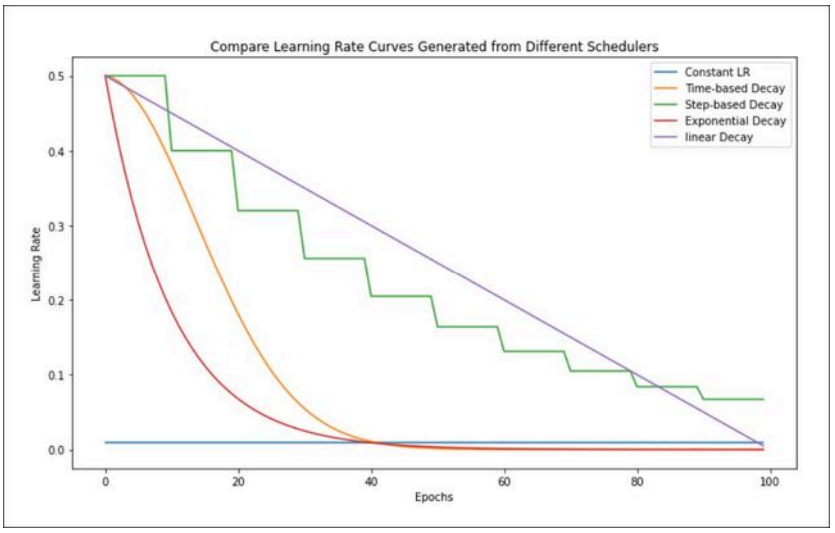


With an example ground truth (i.e., real) image on the left, an indication of autoencoder attention throughout the training process. Second from left, we see that the system is learning the broadest possible lineaments of the identity; the middle picture shows a more advanced stage of training, where the internal semantics of the face are established; second from right, detail comes into focus; and, right, a greater acuity of detail is finally obtained.

At every stage, the autoencoder will be using a loss function, chosen by the user, which will estimate how correct the predictions are. At the early stages, the system will be learning quite quickly, because it needs to obtain 'sketchy' initial information; therefore the learning rate will be set quite high, and the basic structure of the face will quickly emerge (usually inside 30-40 minutes).

Learning Rate

As the system gains more knowledge about the face, the learning rate will need to be adjusted downwards to allow the framework time to study the features that have been extracted from the source material (which slowing-down procedure can either be instigated manually or through an automated learning rate schedule).



We can see from the 'steps' in this visualization of increasing loss (that's a good thing!) during training that the settings have been periodically changed, to encourage ever more profound loss values, so that the model's predictions become increasingly accurate. It's quite clear at what point the learning rate has been changed. The more granular the learning rate schedule, the less 'staccato' these steps will be. Source: https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler

The objective is to reach the *minima* – the point at which the loss function has reached its maximum efficiency on the data. Getting the evolution of the learning rate right is critical for this mission; if the rate is too high, the training processes will 'overshoot' the data, and miss critical details – which will be impossible to recover later, as they are 'foundational' material.

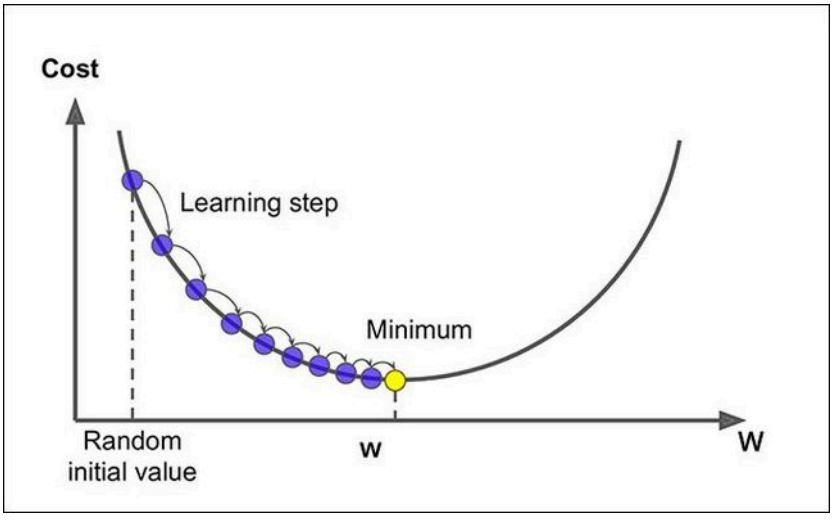
On the other hand, if the learning rate is too low near the beginning of training, the system will begin to 'pick over' as-yet unformed features, before they are ready for that level of attention; in this case too, a non-optimal result is certain.

Gradient Descent

Whatever the current learning rate, the loss function acts as an eager worker whose progress is overseen and controlled by a gradient descent algorithm. The job of the gradient descent algorithm is to regulate and optimize the loss function, by measuring the difference between the values that are being predicted and the genuine values in the real-world data.

When the loss is at its lowest point, the model can be said to have reached convergence – a state in which it is essentially 'complete', and as functional as it is likely to become under the architecture's settings, and with the data that it has to work with.

The gradient descent algorithm continuously updates the model parameters, so that the loss function is increasingly accurate, and makes ever-better predictions. If this sounds like the discriminator function in a Generative Adversarial Network (GAN), however, it isn't, because the loss function has full access to the original data, whereas the adversarial nature of a GAN withholds this from the Generator function.



The nadir of this graph is the optimal state for the trained machine learning network – where the loss between its predictions and the characteristics of the real training data have become almost vanishingly small. Here we see the loss exploding upwards again (the curve up on the right of the picture) after this point, a rare occurrence – but enough of a possibility to make frequent checkpoint saves a necessity. Source: analyticsvidhya.com

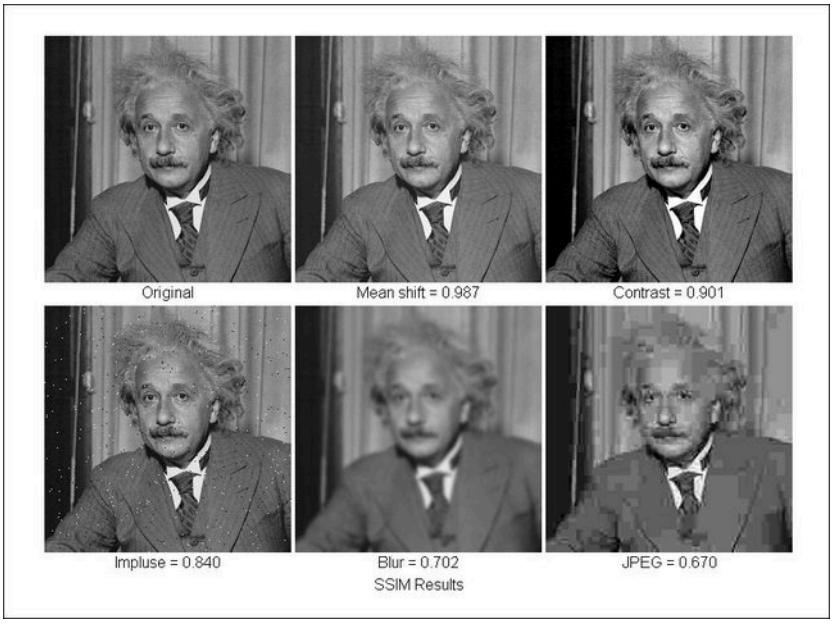
There are a wide variety of gradient descent algorithms, though a handful predominate. Those found commonly in popular image synthesis applications include Adaptive Moment Estimation (Adam, including spin-offs such as Adamax and NAdam, among many others), and Nesterov Momentum.

Popular Loss Functions

CONTINUED ON NEXT PAGE

Structural Similarity Index (SSIM)

SSIM works by comparing the original (source data) image and an image generated by the algorithm during training (such images are automatically generated for evaluation purposes, and these incidences can number in the hundreds of thousands, even in millions of 'disposable' test simulations, which only exist momentarily in VRAM, and from which the only lasting evidence is the extracted loss score).



Here SSIM evaluates the difference between source imagery (upper left) and various possible perturbations, and will return an apposite loss value. Source: https://nsf.gov/news/mmg/mmg_disp.jsp?med_id=79419&from=

SSIM is derived from work released in 2002 by student researchers at IEEE. The work offered a 'new universal objective image quality index', based on three potential loss factors: correlation, distortion in luminance, and distortion in contrast. SSIM is among the last of the purely algorithmic loss functions, in that it was not calibrated via human evaluation like later outings such as LPIPS (see below).

In fact, SSIM's purely math-based approach has since come in for scrutiny and criticism, in comparison to the more human-centric loss functions. Nonetheless, not least because of its deep embedding in the research sector, SSIM remains an image synthesis stalwart.

Mean Absolute Error (MAE)

There are two expected 'default' loss functions in the open source deepfake package FaceSwap – SSIM and MAE.

MAE is popular in many deepfake applications because it's quite robust to outliers, and won't skew the general run of loss values if it sees an anomalous or piece of data. By analogy, MAE won't let outstanding or remedial pupils 'blow the grade curve'.

This is particularly valuable for identity-swapping neural synthesis applications, since the facial data from historical subjects (such as movie stars that passed on over half a century ago, or for whom limited data is available) can rarely be improved or increased in volume. One needs to make the best of what there is, leading to datasets with very high variations in quality.

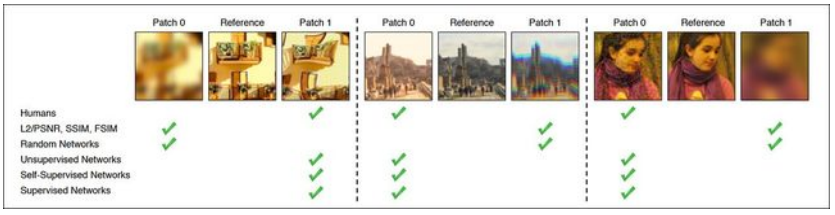
MAE works by measuring the average magnitude of the errors between the genuine values from the source dataset and the predicted values, notwithstanding whether that direction is generally 'up' or 'down'.

https://www.youtube.com/watch?v=fk7bzKFDmk8

As one of the most-used loss functions, MAE's shortcomings have also been variously criticized across the research sector, with some believing that MAE's noise-resistance comes at the cost of its ability to learn relevant patterns. In practice, used, for instance, in a deepfake training session, this could mean that the target identity reconstruction is thorough and workmanlike, but not actually accurate to the source. Thus projects such as improved IMAE have arisen to address these perceived shortcomings.

Learned Perceptual Image Patch Similarity (LPIPS)

The LPIPS loss function, launched in 2018, operates not by comparing 'dead' images with each other, but by extracting features from the images and comparing these in the latent space, making it a particularly resource-intensive loss algorithm. Nonetheless, LPIPS has become one of the hottest loss methods in the image synthesis sector.



LPIPS makes extensive use of human-scored evaluation in order to reach optimal loss functionality. Source: https://arxiv.org/pdf/1801.03924.pdf

Once features have been extracted from the original and predicted image, these are compared using Euclidian distance or cosine similarity. The returned value is backpropagated through the network architecture and the parameters updated, whereupon the whole procedure repeats until convergence is estimated to have been reached, and the training is stopped.

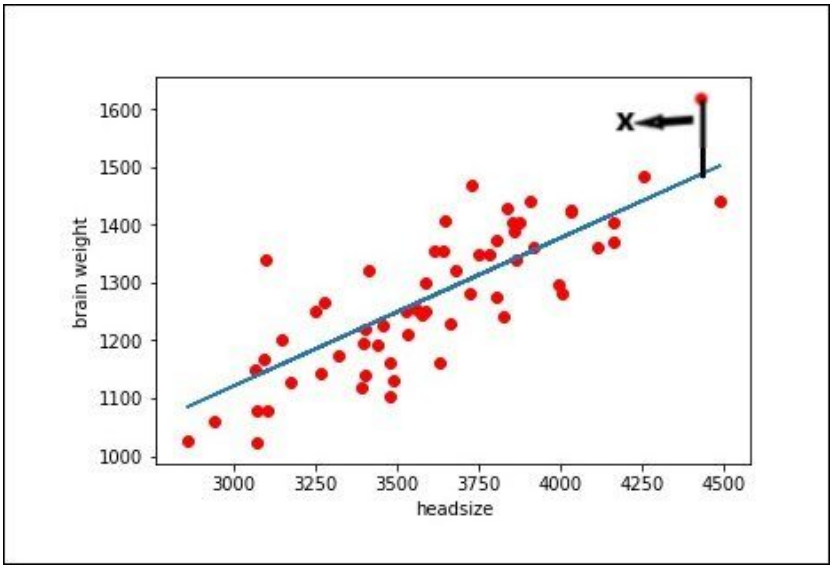
LPIPS is predicated on the notion that human perception regarding image similarity is more consistent and reliable than traditional methods such as MSE (see below) or SSIM. Thus, LPIPS is far from purely mathematical: its development involved the generation of a dedicated dataset containing 484,000 human evaluations around perceived distortion of images. It can be argued that therefore this particular loss algorithm is enormously 'opinionated', and accounts for human psychology in a way that purely mathematical image comparisons cannot match. By the same token, it's logically possible that biases and errors in perception have become enshrined in the algorithm.

Various papers have concluded that LPIPS may offer a more accurate method of comparing images. In practice, it can produce grid-like artefacts during training on image synthesis systems, though these usually diminish as training continues:

One pitfall of LPIPS is its tendency to impose a grid-like structure on reconstructions, a defect that usually either trains out entirely, or to so minimal an effect that it can't be perceived. Source: https://discord.com/channels/441989398465085440/553656548413538314/988240373807743036

Root/Mean Squared Error (RMSE/MSE)

MSE and RMSE are closely related, but function slightly differently. MSE returns the average squared distance between the predicted values and the real values obtained from the data, with lower numbers indicating a better fit. Conversely, RMSE returns the *square root* of the average squared distance, in the same conditions.



RMSE will take into account data points that are pretty far outside the average distribution, which can make for skewed losses in machine learning models that are trained on data of variegated quality. Source: https://www.includehelp.com/ml-ai/root-mean-square%20error-rmse.aspx

While MAE (see above) is seen by many as a more advanced replacement for R/MSE, there are numerous variations on this general concept, including Sum of Squared Error (SSE) and Mean Bias Error (MBE), as well as scale-independent variants such as Mean Percentage Error (MPE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Scaled Error (MASE).

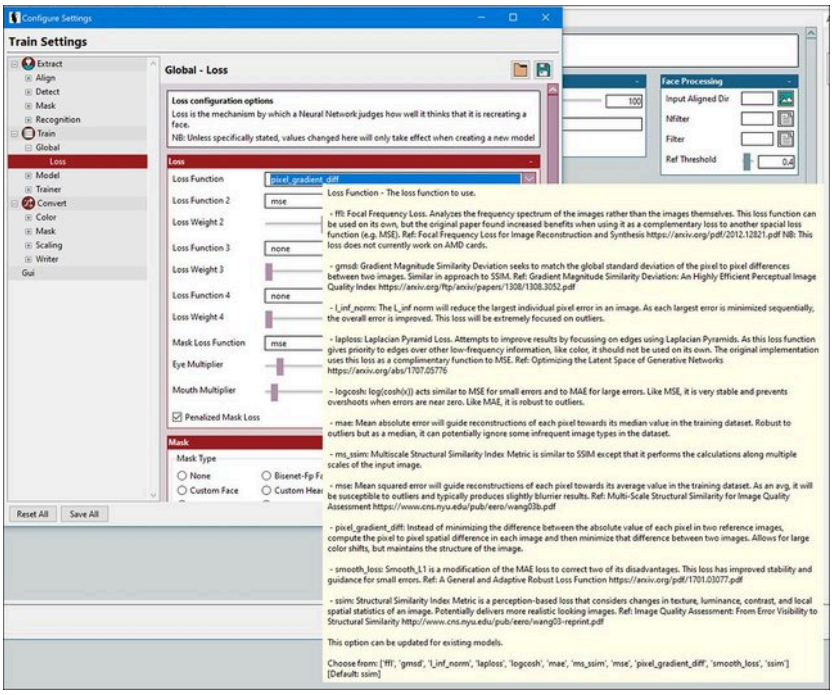
The main reason why alternative loss functions such as those listed above (as well as Huber loss and quantile loss) have overtaken this once-dominant approach is that MSE-style functions can be overly sensitive to outlier data, which is a problem in a research culture that is increasingly seeking to bring in uncurated data at scale and let the algorithms sort out the value of the individual pieces of data.

Thus MSE comes at some cost, in terms of the bias-variance trade-off (how well a model generalizes vs. how well it's able to account for outlier data, because it's easy to get a good generalization when all the data is consistent and well-balanced, though this is not a 'real world' scenario).

A loss function that changes the curve because it sees something that's a little out-of-the-ordinary will likely require extensive (and expensive) data curation and pre-processing; and this doesn't meet the needs of the moment.

Other Loss Functions

Many other loss functions of interest to the image synthesis sector are incorporated into the remarkably comprehensive FaceSwap, though some are obscure and under-appreciated. These include Gradient Magnitude Similarity Deviation (GMSD), Focal Frequency Loss (FFL, particularly aimed at image synthesis systems), Laplacian Pyramid Loss (LPL), and Multi-Scale Structural Similarity, among many others.



Loss functions available in the FaceSwap open source deepfake application.