# Exploring Deep Learning Systems Behavior by Linear Weight Interpolation at Layer Level

Giorgi Merabishvili

Technical University of Munich/Fortiss

## Abstract

Linear weight interpolation at layer level is a method that can generate inputs that will reveal the misbehaviors of classifier. In this report, I will discuss the advantages and results of new techniques. This approach offers smooth transitions between images, increased precession and overall better control over input generating process. Additionally, these techniques minimize the distance between frontier pairs, enhancing the quality and validity of the generated images. Method makes it possible to go very close to the decision boundary and even find images on exact decision boundary where two classes have exactly 0.5 confidence.

## Linear Weight Interpolation

Linear weight interpolation helps changing feature layers gradually and makes it possible to make distance between frontier pairs very low to keep the label after mutation.

$$w_{interpolated} = (1 - \alpha) \cdot w_{source} + \alpha \cdot w_{target}$$

$$0 < \alpha < 1$$

The adjustable $\alpha$ offers significant flexibility in controlling the degree of blending latent vectors, enabling fine-tuned and customizable style mixing. Linear weight interpolation demonstrated better results than previous methods as it can go very close to decision boundary and even find images on exact decision boundary. While linear weight interpolation gives better results, it can be computationally expensive depending on limit of iterations of binary search to find optimal alpha.

## Search for optimal $\alpha$

After checking the seed for initial acceptance, search algorithm starts. Two latent vectors are blended with an interpolation factor – $\alpha$. Algorithm involves searching for optimal interpolation factor alpha which can reveal the misbehavior of the classifier. After every step of binary search, alpha value is passed to linear weight interpolation along with source class and target class vectors. From this formula interpolated vector is passed to synthesis network to generate interpolated image. This image is checked through classifier and based on the softmax output binary search is updated to approach decision boundary where confidence = 0.5. After the search limit is reached two images are saved along with their metadata making up frontier pair: image that caused misbehavior and last image during iterations that was correctly classified. After, measuring the distance between correctly classified and incorrectly classified images, it turns out that L2 distance between frontier pairs is ≈1 and SSIM ≈1.0(iteration limit is set to 20 by default). On 5th page there is a graph that shows first five generated frontier pairs (graph 8). Every pair on the graph has been tested externally and showed L2 distance of ≈ 1 and SSIM of ≈ 1.0.

## Algorithm

**Algorithm** Input and Output Sections

**Input:**
- $D$: deep learning model,
- $class\_source$: source class label,
- $w0\_seeds$: list of initial seeds,
- $\alpha_{max}$: maximum interpolation factor,
- $trunc\_max$: maximum truncation value (0),
- $S_{max}$: maximum search iterations,
- $confidence\_target$: target confidence (e.g., 0.5 for decision boundary)

**Output:**
- $\mathcal{F}$: set of frontier pairs (boundary images near decision boundary)

**Algorithm** Linear Weight Interpolation and Truncation

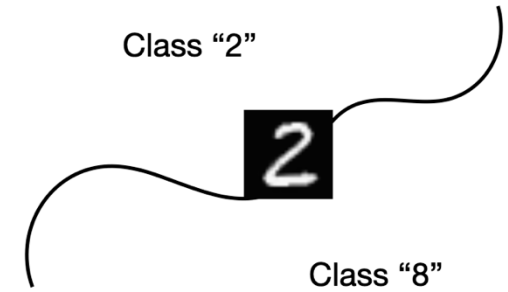
$\mathcal{F} \leftarrow \emptyset$ {Initialize empty set of frontier pairs}
**for** each seed $w0\_seed$ in $w0\_seeds$ **do**
  Generate source latent vector $w$ for $w0\_seed$
  Synthesize image $\mathcal{I} \leftarrow \text{SynthesizeImage}(D, w)$
  Predict class $p \leftarrow \text{Classifier}(D, \mathcal{I})$
  **if** $p \neq class\_source$ **then**
    {Refine with truncation}
    Set $trunc \leftarrow 1$
    **while** $trunc < trunc\_max$ **do**
      Adjust $w$ with truncation level $trunc$
      Synthesize truncated image $\mathcal{I}_{trunc} \leftarrow \text{SynthesizeImage}(D, w)$
      Predict class $p \leftarrow \text{Classifier}(D, \mathcal{I}_{trunc})$
      **if** $p = class\_source$ **then**
        {Accept $w$ and proceed to interpolation search}
        **break**
      **end if**
      Increase $trunc$
    **end while**
    **if** $trunc$ is maxed and $p \neq class\_source$ **then**
      **continue** to next seed
    **end if**
  **end if**
  {Initialize binary search for optimal alpha}
  **while** $iter < S_{max}$ and $\alpha < \alpha_{max}$ **do**
    Interpolate: $w_{interp} \leftarrow \alpha \cdot w2 + (1 - \alpha) \cdot w$
    Synthesize image $\mathcal{I}_\alpha \leftarrow \text{SynthesizeImage}(D, w_{interp})$
    Predict class and confidence $p, conf \leftarrow \text{Classifier}(D, \mathcal{I}_\alpha)$
    **if** $p \neq class\_source$ and $conf \approx confidence\_target$ **then**
      {Store $\mathcal{I}_\alpha$ as boundary image and set $found\_frontier \leftarrow$ True}
      Save previous image $\mathcal{I}_{prev}$ to form a frontier pair
      **break** search for this seed
    **end if**
    **if** $p = class\_source$ **then**
      {Adjust $\alpha$ using binary search}
      **if** $conf < confidence\_target$ **then**
        Increase $\alpha$
      **else**
        Decrease $\alpha$
      **end if**
      Set $\alpha_{step} \leftarrow \alpha_{step}/2$ to refine search
    **end if**
  **end while**
  {Perform additional search to find exact $\alpha$ where $conf = confidence\_target$ (0.5)}
  Add frontier pair $(\mathcal{I}_\alpha, \mathcal{I}_{prev})$ to $\mathcal{F}$
**end for**
**return** $\mathcal{F}$

## Using advantage of swapping while interpolating

Interpolating over every seed would be very insufficient as most of them cannot be “triggered”. Therefore, before starting binary search for optimal alpha value entire layer is swapped which maximizes chances of making changes in the image required to trigger the classifier. In this case interpolation factor lambda is set to 1, meaning it will fully swap the layer. After swapping the layer it checks if the confidence is still over 0.5. If it is over 0.5, it proceeds to the next style mixing operation. If it is below 0.5 it means seed can be triggered with that specific layer blending operation and search starts for alpha.

## Finding Exact Decision Boundary

Linear weight interpolation performed very well in approaching decision boundary and even finding the exact decision boundary. Same algorithm is used to find exact boundary where both source class and target class have confidence of 0.5.(exact boundary was found for every frontier pair in the experiment)



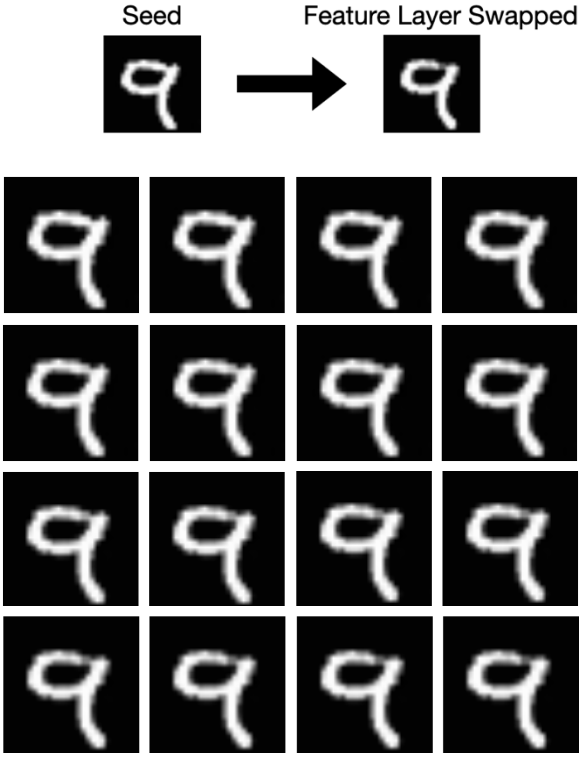


```
{
    "accepted": true,
    "alpha": 0.4063282012939453,
    "base_channel": 0,
    "class_idx": 2,
    "exp-confidence": "0.5000000000000000000",
    "force_fp32": false,
    "img_normalize": true,
    "img_scale_db": 0,
    "input_transform": null,
    "l2_distance": 295.9966383754187,
    "l2_norm": 2523.1254804986597,
    "layer_name": null,
    "mixclass_idx": null,
    "noise_mode": "random",
    "patch_idxs": null,
    "predicted-class": 2,
    "predictions": [
        0.0,
        0.0,
        0.5,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.5,
        0.0
    ],
    "random_seed": 0,
    "sel_channels": 3,
    "ssi": 0.9729326249795006,
    "stylemix_idx": [
        4
    ],
    "stylemix_seed": 0,
    "to_pil": true,
    "trunc_cutoff": null,
    "trunc_psi": 1.0,
    "untransform": false,
    "w0_seeds": [
        [
            32,
            1.0
        ]
    ],
    "w_load": null
}
```


Decision boundary between source class “2” and target class “8”. Graph 1

## Pixel-wise Interpolation

Another method of interpolation is pixel-wise which can be beneficial in terms of speed and saving computational power. On the graph below interpolation between seed and feature layer swapped image is performed with steps shown on the graph.



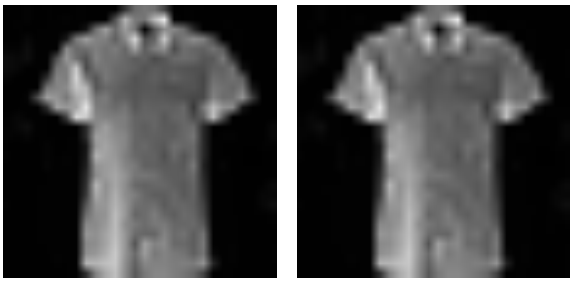

Pixel-wise interpolation between feature different images.
Graph 2

## Pixel-wise vs Vector-wise Interpolation

Vector-wise Interpolation involves interpolating between the latent vectors w1 and w2 at a specific layer using alpha. At every step of binary search new interpolated vector is computed using new alpha and then passed to synthesis layer to generate new interpolated image. Since this method requires generating new image at every step it is computationally heavy. Alternatively, in pixel-wise interpolation method feature layer is fully switched resulting in having two feature different images(graph??). We pass those two images to search algorithm where we do linear weight interpolation directly to the pixel values. In this method, images are generated before search starts and there is no need to generate new images at every step which makes this method faster.

## Mixing Coarse Layers

In StyleGAN we have coarse layers responsible for high-level aspects of an image (pose, shape), middle layers for the smaller scale features (colors, lighting, background) and fine layers for small details (material). Since linear weight interpolation controls the degree of mixing, it results in seamless transitions while mixing even coarse layers. Setting high iteration limit makes it possible to get almost identical pairs even when mixing only coarse layers. Mixing all the layers including coarse layers can be useful to generate diverse frontier pairs and test robustness of Deep Learning system. Frontier pair shown on a graph 1 is a result of first layer mixing. L2 Distance between the frontier pair shown on graph 1 is 14 and SSIM is 0.9998.



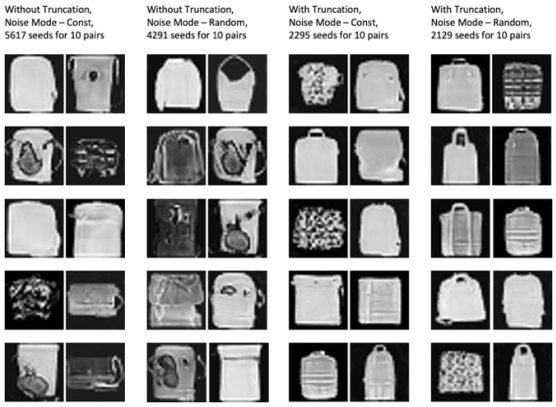
F-MNIST, class 6, interpolation of coarse layer
Graph 3

## Effects of Truncation on Validity of Generated Images

Even high confidence seeds are sometimes somewhat unrealistic and unclassifiable for human eye. Truncation is helpful to increase validity of generated images. After random vector z is sampled from a normal distribution and then passed through a mapping network to produce an intermediate latent vector w, the average of w vectors is computed to get mean vector. Truncated latent vector is computed with formula, in which $\psi$ – truncation psi has range from 0 – 1. meaning all vectors pushed to mean at $\psi$ = 0 and no truncation is applied at $\psi$ = 1.

$$w_{truncated} = w_{average} + \psi \cdot (w - w_{average})$$

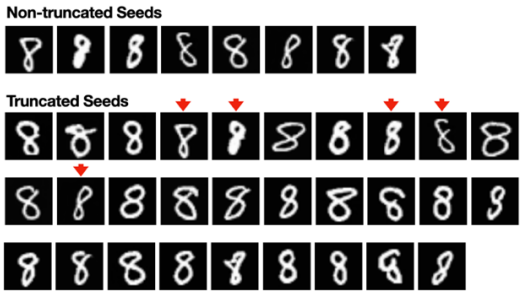
Therefore, by applying truncation images are generated from central region of latent space which ensures to avoid unrealistic images. After observing different settings and combinations, it turned out truncation along with random noise mode increases quality. Noise inputs are injected into each convolution layer of synthesis network. As shown in graph (graph 3, class 8, observed to be the most problematic class in F-MNIST validity wise), truncation works well with random noise mode rather than const noise mode. Graph shows that truncation significantly increased validity and quality of the generated images.




F-MNIST, class 8, truncation and noise mode effects.
Graph 4

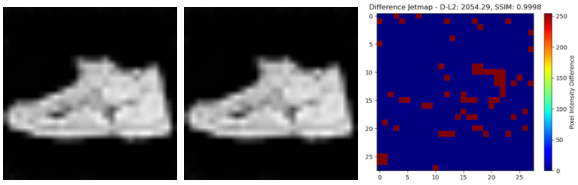
## Truncation Applied in Seed Acceptance

Experiment (graph 4) showed that while generating images truncation increased seed utilization by 2 times. Therefore, if seed fails to get accepted after initial check, truncation is applied and checked again. Note that, without truncation only 8 frontier pairs were generated in the seed limit of 100, while with truncating technique (only initially unaccepted seeds are truncated) it generated 29 pairs. Red arrows indicate seeds that were accepted without truncation.




MNIST, class 8, truncation added (if needed) vs no truncation
Graph 5

## Effects of Background Masking




F-MNIST, linear weight interpolation affecting background
Graph 6

After applying pixel difference heatmaps to frontier pairs, it turned out that while generating frontier pairs background pixels are changed which are completely unnecessary as main focus is the object itself and triggering the classifier by changing up the object is more useful than triggering classifier by making changes in the black background. In some cases, we might allow changing background if dataset is not only focused on one object. Therefore, it depends on the dataset whether we allow changes in background or not. To solve this issue, masking is applied to separate background from the object to only make changes only on the object.

## Acknowledgments

Distances between frontier pairs on graphs are tested externally to ensure there is no mistakes in calculation. For MNIST and F_MNIST dataset vector-wise interpolation and for SVHN pixel-wise interpolation with exact boundary search is set. Note that, pixel-wise interpolation performed better in context of approaching decision boundary.

## Future Work

- Test different metrics in seed validation process to avoid invalid images. After observing rejected and accepted seeds, it seems like it still rejects valid seeds and, in some cases, accepts invalid seeds. Therefore, computational resource is wasted while doing further manipulation with invalid seeds.

- As experiment showed truncation resulted in more stabilized images (graph 3), introducing metric measuring distance between $\omega_{truncated}$ – truncated image/vector and untruncated image/vector might be useful.


```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.50079345703125,
    0.0,
    0.0,
    0.49920654296875,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.49993896484375,
    0.0,
    0.0,
    0.50006103515625,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.5020370483398438,
    0.0,
    0.49796295166015625,
    3.8397492096202255e-38,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.49971771240234375,
    0.0,
    0.5002822875976562,
    3.82726251926432e-38
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.50274658203125,
    0.0,
    0.0,
    0.4972534775733948,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.499237060546875,
    0.0,
    0.0,
    0.500762939453125,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.5031737685203552,
    0.0,
    0.0,
    0.4968262016773224,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.4974976181983948,
    0.0,
    0.0,
    0.50250244140625,
    0.0
],
```



```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.5048216581344604,
    0.0,
    0.0,
    0.4951784014701843,
    0.0
],
```

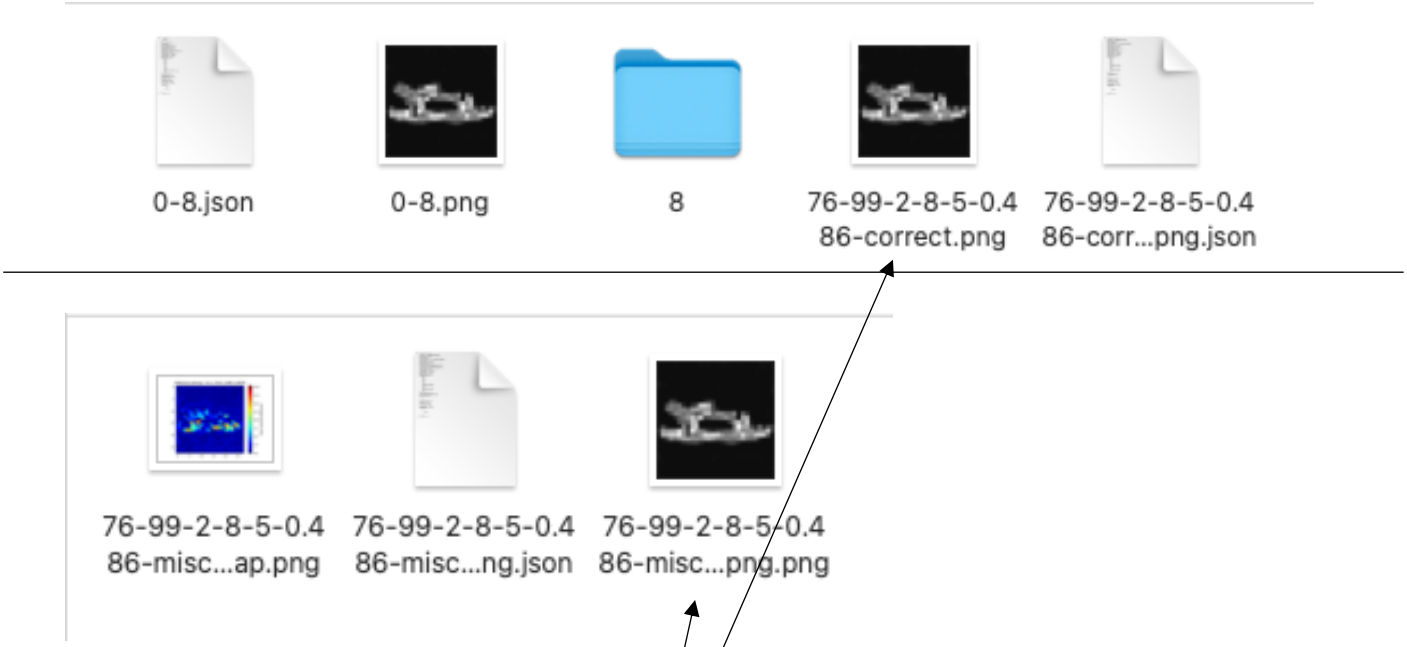


```
"predictions": [
    0.0,
    0.0,
    0.0,
    0.0,
    0.0,
    0.49548354744911194,
    0.0,
    0.0,
    0.5045164823532104,
    0.0
],
```


F-MNIST, class 5, frontier pairs by switching feature layer (graph 2) and then performing pixel-wise interpolation.
Graph 7

In every generated folder there are:

Folder contains seed image and last correctly classified image before misclassification along with their metadata files. Folder “8” contains misclassified image and label “8” indicates that image was misclassified to class “8”.




Generated folder of frontier pair.
Graph 8

Second screenshot shows content of folder “8” – misclassified image and metadata.

Frontier pair in this case are those two images.