Context: Neural Texture Synthesis and Style Transfer

Neural Texture Synthesis (NTS) has two direct applications:

1. Optimizing a new texture
2. Training a texture generator

NTS relies on a loss function to capture distance in texture space. The rich space of deep feature activations is typically used to compute distances in. An ideal loss should capture complete stationary statistics of deep feature activations, be simple to implement and tractable to run.

Existing Loss Functions for Neural Texture Synthesis

1. Second-order statistics
   - Gram-Matrix loss $\mathcal{L}_{\text{Gram}}$ [1] captures correlation between feature channels.

2. Full histogram (i.e., any-order statistics)
   - Optimal transport losses capture the full histogram of feature distribution.

3. Mixtures of losses
   - $\mathcal{L}_{\text{Gram}} + \lambda \mathcal{L}_{\text{OT}}$ [5]
   - $\mathcal{L}_{\text{Gram}} + \lambda \mathcal{L}_{\text{OT}} + \lambda \mathcal{L}_{\text{SlicedWasserstein}}$ [9]

Sliced Wasserstein Loss

We introduce the new loss function $\mathcal{L}_{\text{SlicedWasserstein}}$. It does not rely on a subset of feature-activation statistics. Rather, $\mathcal{L}_{\text{SlicedWasserstein}}$ evaluates the sliced Wasserstein distance [9] between n-dimensional histograms of feature activations.

Spatial Constraints

$\mathcal{L}_{\text{SlicedWasserstein}}$ also supports texture synthesis with spatial constraints. The trick is to concatenate a 1D label or periodicity tag to the feature space. Features of a similar tag are naturally grouped in the (n+1)D histogram: we can proceed using $\mathcal{L}_{\text{SlicedWasserstein}}$ in (n+1)D.

Conclusion: No existing loss function is simple to implement, captures relevant statistics completely, and has tractable complexity.

Paper and code available at bit.ly/2wHMNkJ

All our results are produced using our single loss term.

A Sliced Wasserstein Loss for Neural Texture Synthesis

Eric Heitz, Kenneth Vanhoey, Thomas Chambon and Laurent Belcour