Notes

Advantage/novelty of this algo compared to existing algorithms: real-time (3 times faster than Gatys et al, 1-2 times faster than Chen and Schmidt) and any style (arbitrary), abundant user controls at runtime (?)

Comparison of all style transfer algorithms

|  |  |  |  |
| --- | --- | --- | --- |
|  | Advantages | Disadvantages | Rq |
| Gatys et al | First algorithm to propose style transfer  Good results  Can adapt to arbitrary style | Long (bc of iterative optimization process) |  |
| Forward-feed NN | Fast | cannot adapt to arbitrary styles that are not observed during training (can be up to 300 textures and 16 styles but still a finite set) | replacing slow iterative optimization by fast feed-forward NN that minimizes the same objective |
| Style swap | Faster than Gatys et al but slower than f-f NN | All computational time on style swap | takes feature activations of content and style inputs and then patch by patch replaces the content features with the closest-matching style features |
|  |  |  |  |
|  |  |  |  |

Another question is which loss function to take:

Style loss functions:

* Gram matrix,
* MRF loss [30],
* adversarial loss [31],
* histogram loss [54],
* CORAL loss [41],
* MMD loss [33], and
* distance between channel-wise mean and variance

Rq: feature statistics are necessary for pre-processing dataset such as mean, median, variance, skewness, etc

Mini-batch = when only a subset of available dataset is used

Instant normalization and batch normalization:

Batch normalization scales pictures for each channel across the entire batch. It makes an assumption that images in a batch are representative of the entire dataset.

Instant normalization scales pictures for 1 image at a time across its channels. So it is adapted for cases where each image has its own style (each image is normalized individually). According to authors, instant normalization = feature statistics normalization

Differences between BN and IN:

Another difference is that IN layers are applied at test time unchanged, whereas BN layers usually replace mini- batch statistics with population statistics.

BN resolves issue of internal covariance shift, it makes assumption that a mini-batch is representative of training set which isn’t a case in test set so to improve robustness, we replace mini-batch stats with population stats. However, IN not making such an assumption, stats remain unchanged. BN: during inference statistics are fixed (=those from training set), IN: statistics are calculated during both training and testing.

CIN:

We adapt affine parameters gamma (=scale) and beta (=shift) to each style from a finite set of styles (y^s and b^s in the formula).

AdaIN:

We adapt affine parameters to each individual picture on the fly (=adaptively). Gamma becomes std deviation and beta becomes mean.

Architecture:

Rq: reflection padding = instead of padding with zero or a constant, we pad with values close to that of borders of image to avoid edge effect

Project roadmap:

1. Explain in great detail the paper itself (order to be improved)
   1. Background, previous research
   2. Gap
   3. Solution, interest of AdaIN
   4. Comparison of BN, IN, CIN and AdaIN
   5. Interpretation of AdaIN (style normalization, not just contrast of image)
   6. Why important architectural choices were made (why AdaIN, why reflection padding, why no BN/IN layer in encoder, etc.)
   7. Architecture of AdaIN
   8. Training dataset
2. Use AdaIN ourselves + Comparison with other algo (both in terms of time and performance (=loss) as well as results (visually)) by ourselves
3. Potential applications (and maybe we can implement/apply AdaIN ourselves ?? to show its advantages, why this research was important)
4. Search for loopholes and things to be improved (try to improve??)

??