On a higher level they replace the need for domain or single ~~label~~ image label annotation in Image-to-Image translation by training a guiding network. that is able to do a self clustering of the image domain and therefore that guides the image-to-image translation, instead of the previously needed labels.

Until this paper these I-I translation requires labels because you need to know how to build these domains at the top. to get the different style vectors out of one will actually need label annotation for each image ie for each single image you need to know which one of the source corresponds to which one of the (a,d).

~~Stages~~

~~Corresponding Images → Sketch of a shoe~~

meaning the model doesn't require a labelled pair of images to train the network.

Stages that I-I translation went through

1) Corresponding Images One-to-One.

Here is an example of the sketch of the shoe and here is the corresponding shoe. and similarly with the other shoes..
And from that you could learn a model that translates from one domain to the other because you have corresponding image-level annotations. (~~which~~ basically which element of domain A corresponds to which element of domain B.

2) Set-level Annotation → That is basically the supervised labels for domain.
2 domains. These things are instances of class A and those of class B. Yet there is no correspondence b/w the 2 classes. So I-I translation is not possible b/w domains where we just have domain level labels.

Date: / /201

8. This paper introduced the following where you do not have domains any more but you just have the data set.
Now the hypothesis is that there are still going to be domains in the dataset. They can be overlapping or not but there are still going to be domains, we just dont know what they are. Here there can be many ways to differentiate eg male/female, hair color. etc.
So in ~~essence~~ essence we are going to assume that there are still some kind of a domain structure you just don't know what it is, but if we knew what it was then we could simply apply methods from ~~set~~ -level and the work will be done

Now, what this paper shows is that if we apply something like a self clustering approach on x and then learn y-y translation then ~~this method is not as good as~~ quality the quality will not be as good as if we do both the things jointly.
So, what this paper does is that it jointly learns to cluster (ie self label) the imgs and to do y-y ~~translation~~. ~~that~~ And by doing these both jointly then help each other perform better.

Now how ~~we~~ do they do this?

There are 3 different parts to their model.

1. Encoder → which is the guiding network
2. Generator } So, the generator and discriminator
3. Discriminator } they are the standard GAN generators and discriminators ie General Adversarial Network., but they do have some changes.

First We will see the discriminator.

It gets an image which can either be a generated or a real image. and it needs to decide that whether the image is real or fake.

~~How the input is a vector or image in this~~ domain

Here we can see that input domain is a vector.

~~or for this so. However there are multiple heads~~

This whole thing is built on the pseudo - clustering approach ~~so~~ ie the pseudo labels which are coming from the left side here. In essense we can assume that there are multiple classes in the dataset and the heads or is here corresponds to each class. From somewhere outside, it will get the information of

the classification that what it is suppposed to be. and then there is a head at the top of classifier which discriminates the fake from our classification.

Hence, this discriminator is sort of a conditioned discriminator. conditioned on a label. It is just a labelled conditioned discriminator that discriminates b/w real and fake.

Now how we train the discriminator is that we would give it an image and we would let the encoder label the image (the process for which we will see after this) and for that particular label we will let the discriminator label ~~this~~ ~~one~~ classify the image as real & fake.

~~Now coming to the generator~~

The Encoder what it does is that it will take an image and it will output 2 things.

1) Label ↓ It is suppposed to be a number b/w 0 -- k-1 ie suppposed to be a class label → we just guess the no of classes and we shorten our region a little.

2) Style Code

These two things go on a different pathways The label is directly going to the discriminator ~~The generator does not see the label.~~ and the style code goes to the generator. Now the generator takes a source image and it takes the style code. Now the style code is encapsulating the style of the reference image.

The way we gonna train is that the style is gonna to describe the style somehow all the images that are from this label.

So the generator takes both the things and produce a generated image. The discriminator is then tasked with differentiating that whether the image is real or fake for the given label. over here.

And all of this is trained jointly. If we jointly train the encoder to produce its class labels. and the style. we train the generator to take in the style & the source image and output an image to fool the discriminator and the discriminator at the same time is trained to differentiate b/w real and fake images based on the label that the encoder gives.

At the end of it

↳ we first need a way to train the encoder to come up with suitable class labels even though it doesn't get any back propagation signal into that part of its network.

8:12

So, that's where we come to our loss function: The way we are going to do this is that we are going to take the following approach. We are going to take an image x and we are going to take a randomly augmented version $x^+$. These are basically the ideas from self supervision which is the work of some of the paper

We are going to take an image and we are going to augment it by random cropping, change its luminance etc. So we have two versions of the same image.

And we want to maximise the mutual information in not b/w the images themselves but $p_v$ is going to be the output of the encoder here.

P.T.O

If we say that
X goes into the encoder and the encoder outputs style and label class

P is going to be the class distribution of a histogram, from which we are going to sample the label Y

~~P is the distribution over the output classes~~.

So, since we don't have a label we can't train the distribution. So what we will have to say is that we want to maximize the mutual information b/w the Output Distribution of the Image ($I_P$) and the output distribution of the augmented image. ($I_{P+}$)

That entails on following two parts.
Entropy of P and the conditional entropy of P given P augmented
we want to maximum entropy (P) to maximize of MI.

In simple language what it means is that

Date: / /201

we want different x's let say $x_1, x_2 \dots, x_n$ and we want them to have different distributions in labels.

So, if the entropy is really high of the distribution p that means that different images get assigned to different classes. and if that is low that will mean that all the images get assigned to the single class.

So that's ~~today~~ the first thing that we want to maximise the entropy to maximize ~~or~~ the mutual information. and second we need to minimize the conditional entropy of p given p augmented.

Now, what it means is that. if we know the augmented version of the image, its class labelling should be same as the ~~or~~ unaugmented version; which means that if we take one of the x to $x_i^+$, then that should keep the same class labelling.

~~If~~ a airplane in front of a blue sky is ~~still~~ an airplane in front of a bit blue sky. So that is why they should have the same output.

So by this Mutual Information we sort of get ~~output~~ clustering of the output space.

↳ a rough

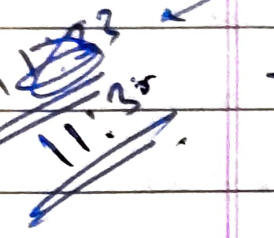
By maximizing the MI, the network is encouraged to distribute all the samples as evenly possible.

That's how we train the pseudo clustering approach.

Now, we have a label to each image.

So now how do we train the other parts.

The other parts are the additional losses here



$P_i$ denotes the $K$-dimensional marginal probability vector

and $P_{ij}$ denotes the joint probability.

Coming to the style Part.

Encoder outputs the labelling we got that covered. and it outputs the style part. Now as we saw earlier the style part goes to the generator and in this part we also do get a generator back propagation to the style part; which implies that our encoder is trained to help the generator with its task of fooling the discriminator.

But first we will look at the loss function of the style loss of the encoder

The style code is the contrastive loss b/w numerator & denominator.

For this let us assume that ~~there~~ if we have our dataset and we take images out and train our dataset.

We take some x batches of images

So, we sort of have a queue. And we keep on pushing the new images into the queue and ~~throw the old image~~ out. So, for each image there ~~are some other images.~~ is always going to be a queue of some other images.

The denominator over here is the others ie the other images.

$s$ → style code of the image you are considering right now

$s^+$ → style code of the augumented image.

Again considering $X_1, X_2, X_3 \ldots$ we have our $X_1$ we put it through the encoder and we get our style code, and we augument $X_1$ to be $X_1^+$ and when we put that through the encoder it gives us $s_1^+$ and we put all the other images to the encoder and they are the $s^-$.

Now what we require is that the style code of our image is closer to the style code of its augumented version ~~than to the previous~~ than it is to any of the other images style code.

So its a contrastive of two things where you pull together things that are close to each other and you push apart things that are we think should be far away from each other.

Here are some other loss functions used in the paper

1) The model uses the adversal loss. This loss is not that different from the loss function used in typical GAN MODELS. The output D denotes the logic from the domain specific discriminator and S-TELDA denotes the target style code vector from the guidance network.

The generator G - learns to translate input image X to the target domain Y - TELDA while reflecting this style code.

In order to prevent degenerate situation when the generator ignores the given style code vector the model also use the style contrastive loss. The loss function guids the generated image to have a style similar to the reference image and dissimilar to random negative samples

Lastly to ensure that the generator can reconstruct its source image x when given with its orginal style s.
The model uses a reconstruction loss. The function not only ensures generator to preserve domain invarient characteristics like pose of its input image but also helps to learn the style representation of the guiding network by embracing the original style of the source image.

The final loss function is shown. The lambdas are the hyperparameter.

Coming to the overall loss function.

1) Discriminator → The discriminator is just the adversarial loss

2) Generator → has the Adversarial loss + The style round trip consistency + its own image round trip consistency.

3) Encoder → All the generator loss + Mutual Info loss + Style Contrastive loss

Comparisons

What if we separately learn the guider network and then translate. →

Result → NS ; t-SNE → non linear.

FID → Quality metrix for GAN's lower is better.