CS 597 – REPORT 1

Yash Pradeep Gupte
MS Computer Science
CWID: A20472798

[I] Dataset – MS COCO 2017– Zebra class

Zebra class consists of total 2001 instances with each image having 5 captions describing it in detail.

[II]Attention GANs - Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks

1) Introduction and Related work

Text to Image synthesis task with Attention mechanism which synthesize fine grained details at different sub regions of the image by paying attention to relevant words in the natural language description. This paper proposed a novel loss for generator called as DAMSM loss (Deep Attentional Multimodal Similarity Model). The model was trained on CUB dataset and COCO dataset.

The motivation for this paper comes from various Text to image synthesis Generative Adversarial Network models proposed in the GAN community. Attention models have shown significant results in machine translation tasks. The encoder decoder architecture is a sequence-to-sequence task which incorporated with GANs describes image generation as an m-stage process.

2) Architecture

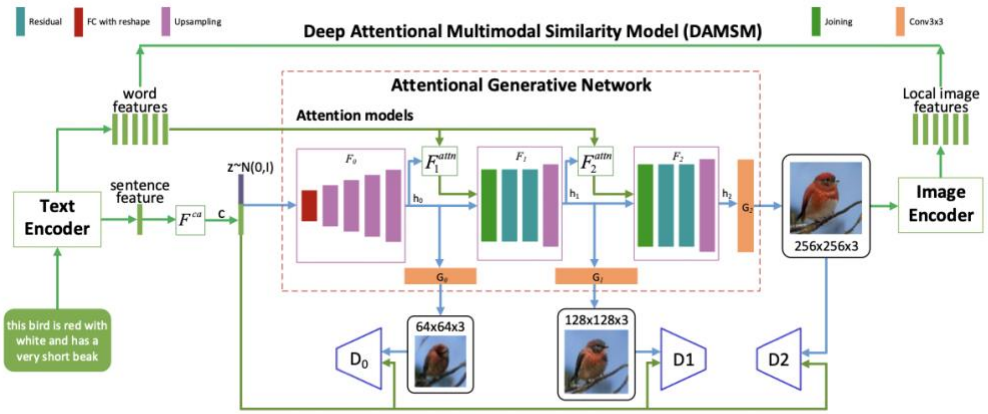

Figure 2. The architecture of the proposed AttnGAN. Each attention model automatically retrieves the conditions (*i.e.*, the most relevant word vectors) for generating different sub-regions of the image; the DAMSM provides the fine-grained image-text matching loss for the generative network.

(A)Text encoder is a bidirectional LSTM which concatenates the hidden state for forward and backward directions

Sentence feature: final hidden state

Word feature: Hidden states from all time stamps

(B) Conditioning Augmentation

- Randomly sample latent variables from Gaussian distribution
- Add Noise (N(0, I)) to have higher variation in generated images even for same caption due to Fca

(C) Generator F0

-Responsible for Upsampling

- Upsampling done with Nearest Neighbor Interpolation (scale factor = *2) that is double height and width

-first stage f0 does not use word level features

(D) Attention Network F1attn

- Combines word features with previous stage context(hi-1) by implementing following steps:
    i. Bring word features to common space
    ii. Combine them with the context
    iii. Generate a word context feature vector for a region
    iv. Repeat for each region

(E) Generator F1

- Responsible for sampling
- Upsampling done with Nearest Neighbor Interpolation (scale factor = *2) that is double height and width
- Combined with ResNetBlocks where height and width is unchanged
- Uses word level features from F1attn

(F) Attention Network F2attn

- Same structure as F1attn
- Uses h1 as input

(G) Final Stage Generator F2

-Same structure as F1 generator

- Uses h1 and output of F2attn as the input

(H) Generators G0, G1, and G2

- Convolutional layers to reduce number of channels to 3
- Losses: each generator has a separate loss
- Total Generator loss is summation of all the generator losses

(I) Discriminators D0, D1 and D2

- Considers only the sentence level features
- Two forms: Unconditional and Conditional
    Unconditional: images are real or fake?
    Conditional: Are the images and caption of the same pair?

(J) Image encoder

- Required to compute the proposed loss (DAMSM)
- This is a standard convolutional network – Inception v3 pretrained on ImageNet

(K) Deep Attentional Multimodal Similarity Model (DAMSM)

- Does generated image actually follow the description?
- The DAMSM loss model can be separately trained as it does not always use generated images.

(L) Architecture loss

- Generator loss + DAMSM loss

3) Experiments

They have trained their model on

- Caltech – UCSD – Birds(CUB) dataset which had longer decriptions – 10 captions per image
- COCO dataset which has smaller descriptions – 5 captions per image

4) Evaluation

Method	inception score	R-precision(%)
AttnGAN1, no DAMSM	$3.98 \pm .04$	10.37 ± 5.88
AttnGAN1, $\lambda = 0.1$	$4.19 \pm .06$	16.55 ± 4.83
AttnGAN1, $\lambda = 1$	$4.35 \pm .05$	34.96 ± 4.02
AttnGAN1, $\lambda = 5$	$4.35 \pm .04$	58.65 ± 5.41
AttnGAN1, $\lambda = 10$	$4.29 \pm .05$	63.87 ± 4.85
AttnGAN2, $\lambda = 5$	$\mathbf{4.36 \pm .03}$	$\mathbf{67.82 \pm 4.43}$
AttnGAN2, $\lambda = 50$ **(COCO)**	$\mathbf{25.89 \pm .47}$	$\mathbf{85.47 \pm 3.69}$

Table 2. The best inception score and the corresponding R-precision rate of each AttnGAN model on CUB (top six rows) and COCO (the last row) test sets. More results in Figure 3.

5) Component Analysis

- DAMSM aims to keep generated image relevant to description
- Attentional GAN – aims to add details at each stage
- AttnGAN1 – image output = 128x128x3

-AttnGAN2 – image output = 256x256x3

[II] TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up

Memory friendly generator and Multi scale discriminator with novel grid self-attention

1) Introduction and Contributions

- Model Architecture
    - Build GAN using purely transformers and no convolutions
    - Discriminator – Vision Transformer
    - Generator – Transformer interlaced with Upsampling

- Training Technique
    - Data Augmentation
    - Multitask Co training for generator
    - Localized initialization for self-attention
- Performance
    - Trans GAN XL – IS score = 8.63 and FID score = 11.89 on CIFAR10 dataset

2) Architecture

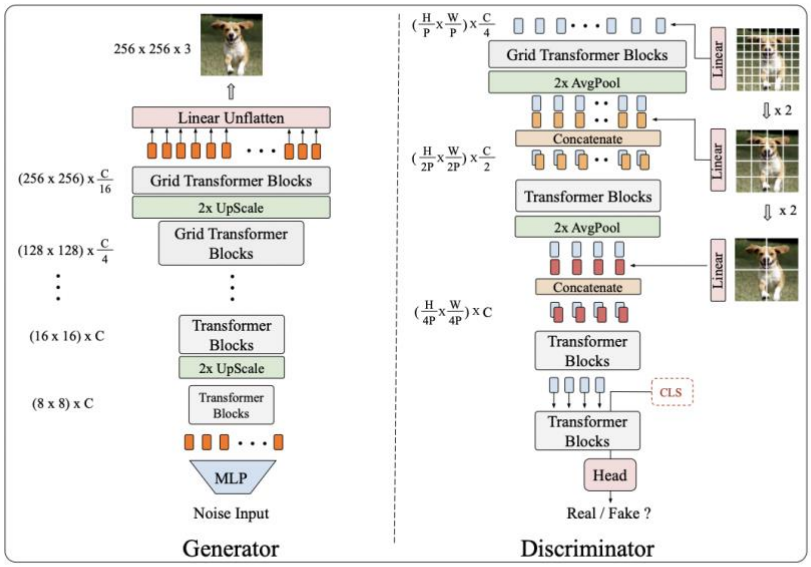

Figure 2: The pipeline of the pure transform-based generator and discriminator of TransGAN. We take 256×256 resolution image generation task as a typical example to illustrate the main procedure. Here patch size p is set to 32 as an example for the convenience of illustration, while practically the patch size is normally set to be no more than 8×8, depending on the specific dataset. `Grid Transformer Blocks` refers to the transformer blocks with the proposed grid self-attention. Detailed architecture configurations are included in Appendix C.

(A) Discriminator

- Vision Transformer (ViT) – Transformer based image classification
- Image is unrolled into subpixels or regions (super pixels)
- Linear flatten – unroll the super pixels into a single vector and that then is like a word in a sentence
- Picture becomes a series of vector and then they applied regular transformer architecture
- Then this goes to a Transformer Encoder
- At the end we classify it as Real or Fake
- They added positional encoding because of lacking convolutions, the transformer has no idea where in the picture a given thing appears.

(B) Generator

- Output of generator is 32x32x3
- Upsampling architecture – the problem with transformers is that they do require quite a bit of memory and compute resources.
- Therefore, we have intrinsic upscaling of dimensions that is at the beginning we have noise input, and we have MLP generating initial sequence (8x8x c)
- noise generator creates an 8x8xc grid
- we get 64 tokens from the 8x8xc grid that is it's like a sentence with words
- we then pass these tokens to a transformer encoder with 'M' layers
- Upscaling from 8x8xc to 16x16x(c/4) – more resolution and less channels
- finally, we end up with target resolution of 256x256x(c/16) which we then feed it to a small linear projection layer i.e. "linear unflatten" layer to get 3 channels.

(C) Grid Self Attention

- Grid Self-Attention across different transformer stages. We replace Standard Self-Attention with Grid Self-Attention when the resolution is higher than 32 × 32 and the grid size is set to be 16 × 16 by default.
- Rather than calculating the connection between a given token and all other tokens, grid self-attention divides the full-size feature map into many non-overlapping grids, with token interactions calculated inside each local grid.
- On high-resolution stages (resolution more than 32 32), we add grid self-attention while keeping conventional self-attention on low-resolution stages.

3) Comparative Analysis

- They have compared their architecture with AutoGAN and as soon as they replace discriminator with the transformer the performance drops drastically
- Hence, they have applied three tricks improve the performance

(A) Data Augmentation is crucial for TransGANs

- They implement data augmentation from a paper called "Differential Augmentation for Data Efficient GAN Training". Where the augmentation T is a differentiable function.
- Given that Transformers do not have convolutions (locality biased built into their architecture), they need a lot more data. Transformers work well if there is a lot of data.
- Adding Data augmentation in GANs improves the performance drastically.

(B) Co-training with self-supervised auxiliary task

- This step performs a Super Resolution Task
- In addition to GAN training, we have the dataset and those images are fed to the discriminator along with the generated images (this is the main GAN loss)
- We also take the dataset images and feed those to the target (SR). This is the target for the GAN.
- So, the GAN needs to output something and what it gets as an input is (LR) which is scaled down to (SR)
- We take images from dataset and deliberately down sample them (LR) and then task of GAN is to predict the high-resolution SR images.

4) DATASET

- They start with three common datasets – CIFAR 10, STL-10 and CelebA
- CIFAR10 - 60k 32 × 32 images, with 50k training and 10k testing images
- STL-10 - 5k training images and 100k unlabeled images, and all are resized to 48 × 48 resolution.
- CelebA - 200k unlabeled face images (aligned and cropped version), with each image at 128 × 128 resolution
- Further they considered, CelebA - HQ (30k images) and LSUN Church (125k images) all of 256 × 256 resolution

5) RESULTS

Methods	**CIFAR-10**		**STL-10**		**CelebA**
	IS↑	FID↓	IS↑	FID↓	FID↓
WGAN-GP [1]	6.49 ± 0.09	39.68	-	-	-
SN-GAN [46]	8.22 ± 0.05	-	9.16 ± 0.12	40.1	-
AutoGAN [18]	8.55 ± 0.10	12.42	9.16 ± 0.12	31.01	-
AdversarialNAS-GAN [18]	8.74 ± 0.07	10.87	9.63 ± 0.19	26.98	-
Progressive-GAN [16]	8.80 ± 0.05	15.52	-	-	7.30
COCO-GAN [66]	-	-	-	-	5.74
StyleGAN-V2 [68]	9.18	11.07	10.21* ± 0.14	20.84*	5.59*
StyleGAN-V2 + DiffAug. [68]	**9.40**	9.89	10.31* ± 0.12	19.15*	5.40*
TransGAN	9.02 ± 0.12	**9.26**	**10.43** ± 0.16	**18.28**	**5.28**

Future plans:

-Execute Transformer GAN implementation of Xu