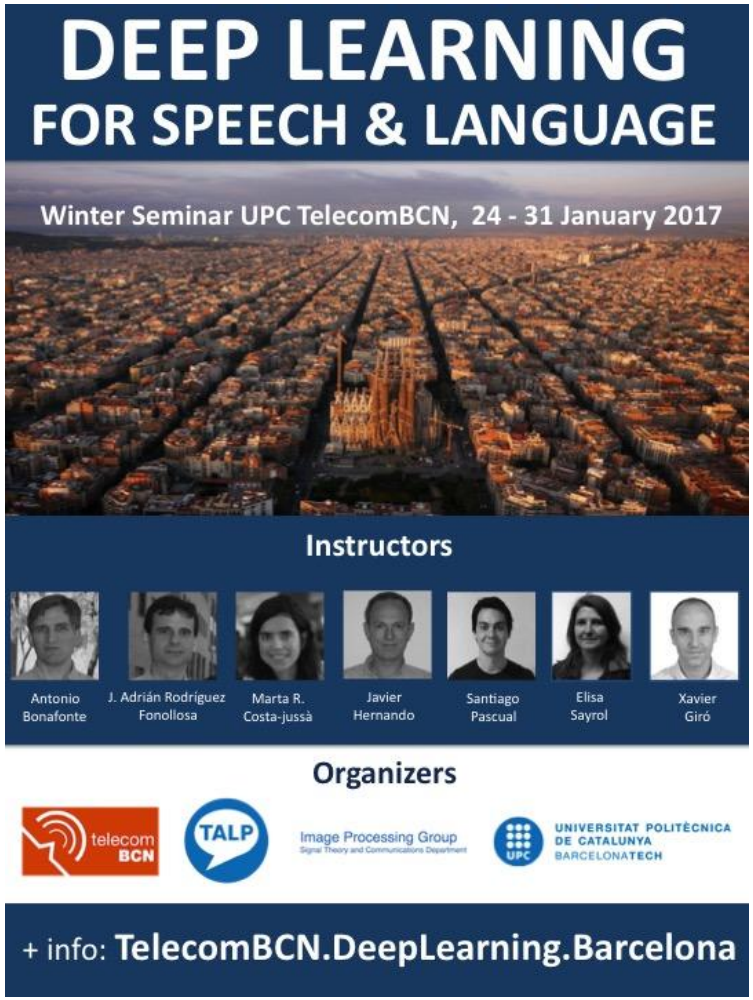


[course site]

Day 2 Lecture 5

Generative Adversarial Networks

Santiago Pascual

TALP UPC

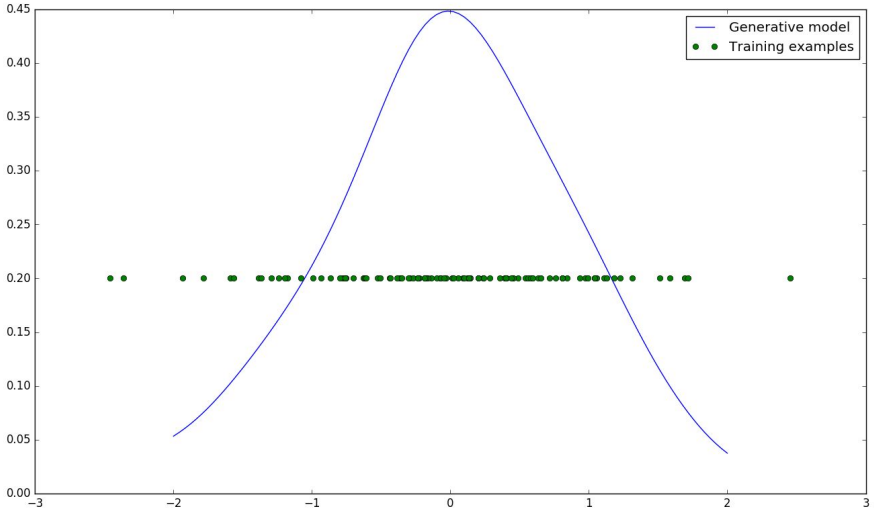
Outline

1. What are Generative models?
2. Why Generative Models?
3. Which Generative Models?
4. Generative Adversarial Networks
5. Applications to Images
6. Applications to Speech

What are Generative Models?

We want our model with parameters $\boldsymbol{\theta}$ = *{weights, biases}* and outputs distributed like *Pmodel* to estimate the distribution of our training data *Pdata*.

Example) y = f(**x**), where y is scalar, make *Pmodel* similar to *Pdata* by training the parameters $\boldsymbol{\theta}$ to maximize their similarity.



What are Generative Models?

Key Idea: our model cares about what distribution generated the input data points, and we want to mimic it with our probabilistic model. **Our learned model should be able to make up new samples from the distribution**, **not just copy and paste existing samples**!

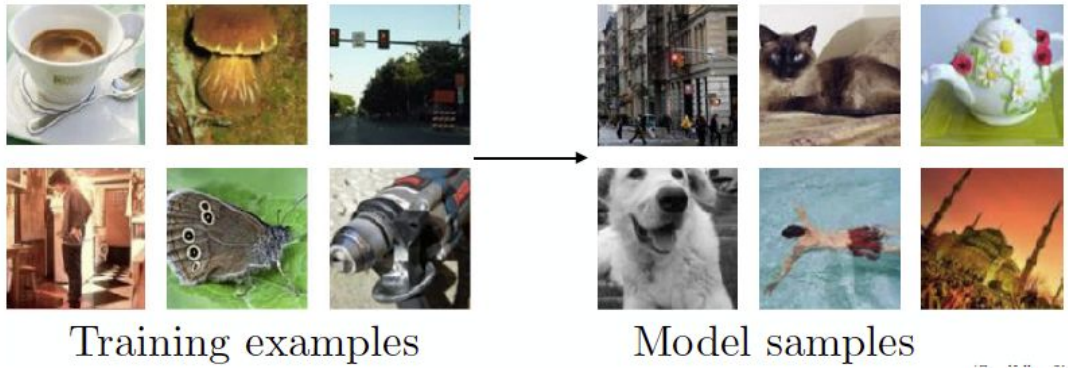


Figure from [NIPS 2016 Tutorial: Generative Adversarial Networks (I. Goodfellow)]

Why Generative Models?

- Model very complex and high-dimensional distributions.
- Be able to generate realistic synthetic samples
  - possibly perform data augmentation
  - simulate possible futures for learning algorithms
- Fill the blanks in the data
- Manipulate real samples with the assistance of the generative model
  - Edit pictures (exemplified later)

Current most known options:

1. Generative Adversarial Networks
2. Autoregressive methods (WaveNet) → stay tuned, next chapters
3. Variational Autoencoder

Generative Adversarial Networks (GAN)

We have a pair of networks, Generator (G) and Discriminator (D):

- They “fight” against each other during training→ **Adversarial Training**
- G mission: make its pdf, *Pmodel*, as much similar as possible to our training set distribution *Pdata* → Try to make predictions so realistic that D can’t distinguish
- *D* mission: distinguish between *G* samples and real samples

Adversarial Training (Goodfellow et al. 2014)

We have networks **G** and **D**, and **training set with pdf *Pdata***. Notation:

- ***θ(G)***, ***θ(D)*** (Parameters of model **G** and **D** respectively)
- ***x*** ~ Pdata (M-dim sample from training data pdf)
- ***z*** ~ *N(0, I)* (sample from prior pdf, e.g. N-dim normal)
- **G(z) = *ẍ*** ~ Pg (M-dim sample from G network)

D network receives **x** or **ẍ** inputs → decides whether input is real or fake. It is **optimized to learn: x is real (1), ẍ is fake (0) (binary classifier).**

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right).$$

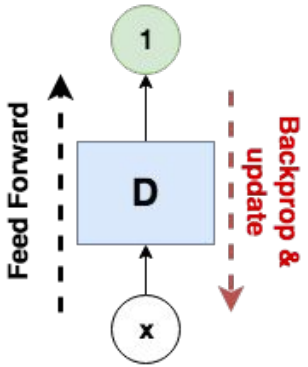
G network maps sample **z** to G(**z**) = **ẍ** → it is **optimized to maximize D mistakes.**

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

NIPS 2016 Tutorial: Generative Adversarial Networks. Ian Goodfellow

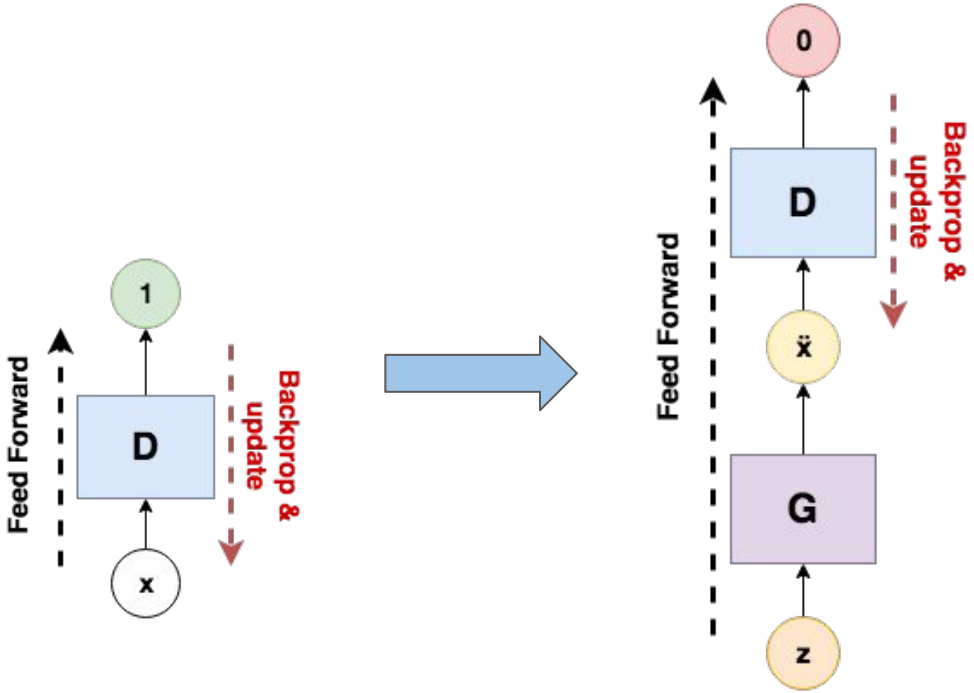
Adversarial Training (batch update)

- Pick a sample ***x*** from training set
- Show ***x*** to **D** and update weights to output 1 (real)



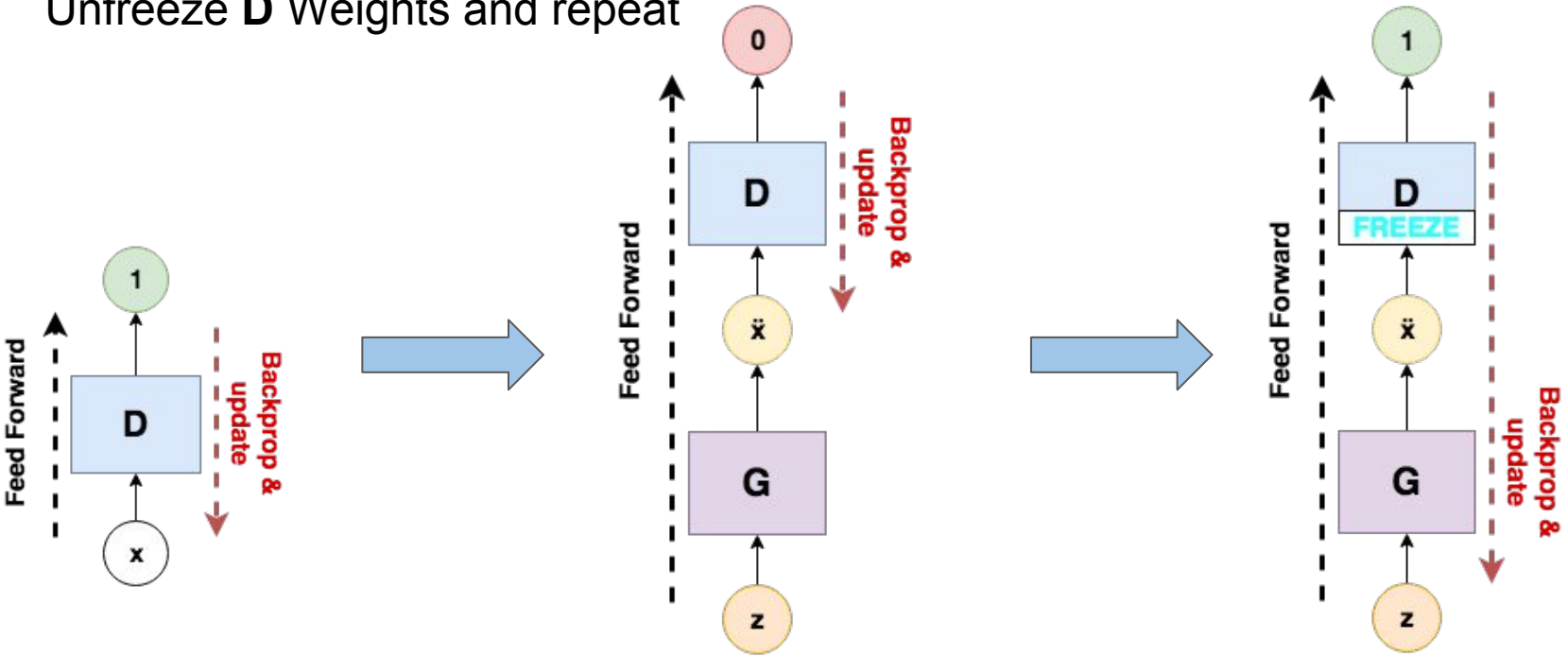
Adversarial Training (batch update)

- **G** maps sample ***z*** to ***ẍ***
- show ***ẍ*** and update weights to output 0 (fake)



Adversarial Training (batch update)

- Freeze **D** weights
- Update **G** weights to make **D** output 1 (just **G** weights!)
- Unfreeze **D** Weights and repeat



Adversarial Training analogy

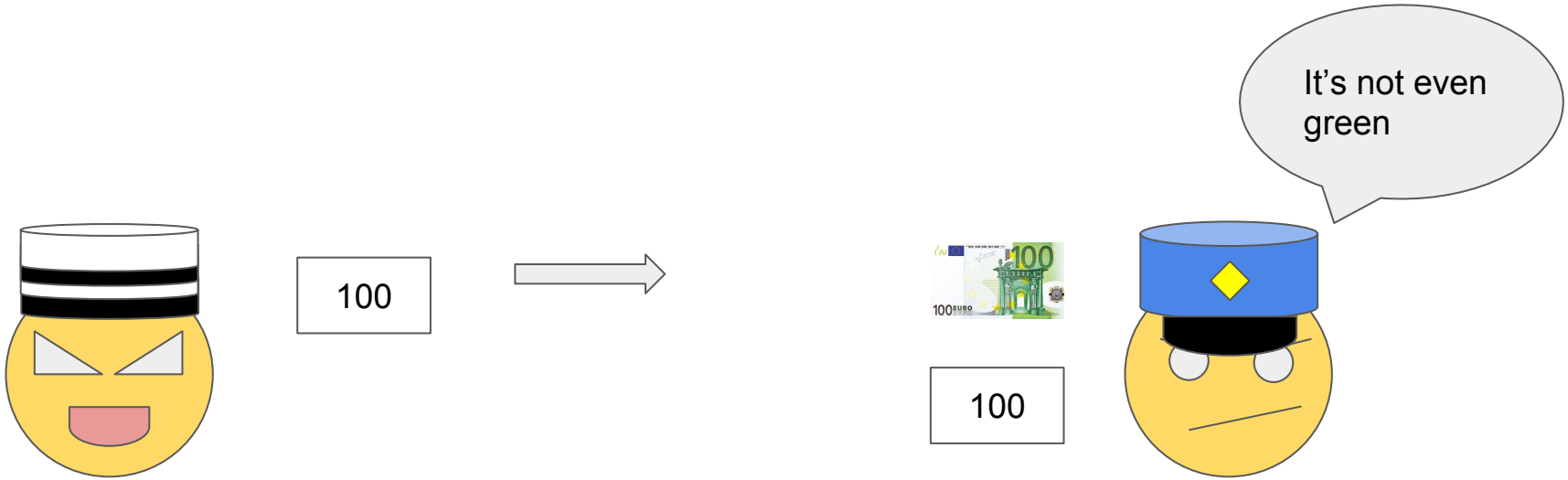
Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.

Key Idea: as **D** is trained to detect fraud, its parameters learn discriminative features of “what is real/fake”. As backprop goes through **D** to **G** there happens to be information leaking about the requirements for bank notes to look real. This makes **G** perform small corrections by little steps to get closer and closer to what a real sample would be.

- **Caveat:** this means GANs are not suitable for discrete tokens predictions, like words, because in that discrete space there is no “small change” criteria to get to a neighbour word (but can work in a word embedding space for example)

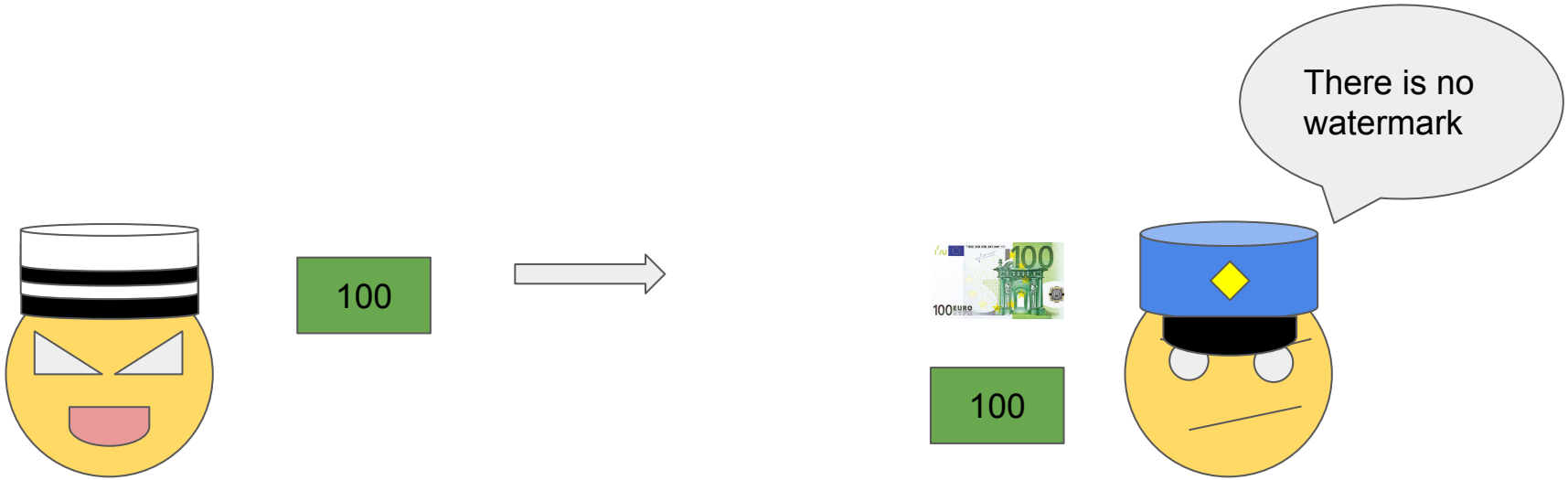
Adversarial Training analogy

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.



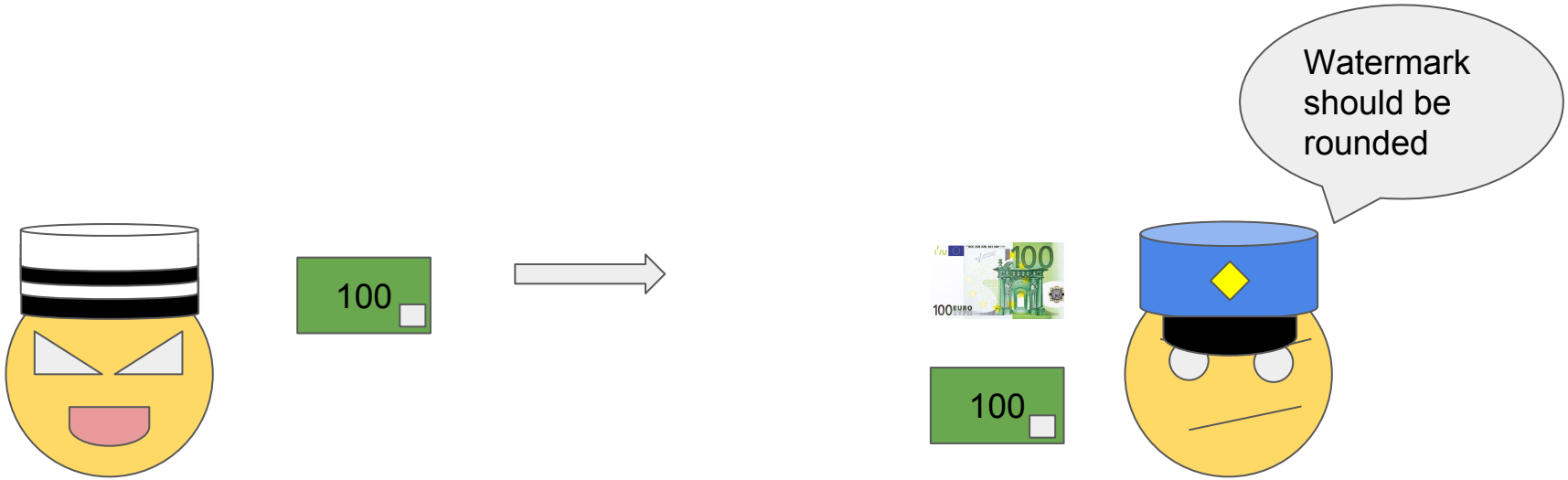
Adversarial Training analogy

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.



Adversarial Training analogy

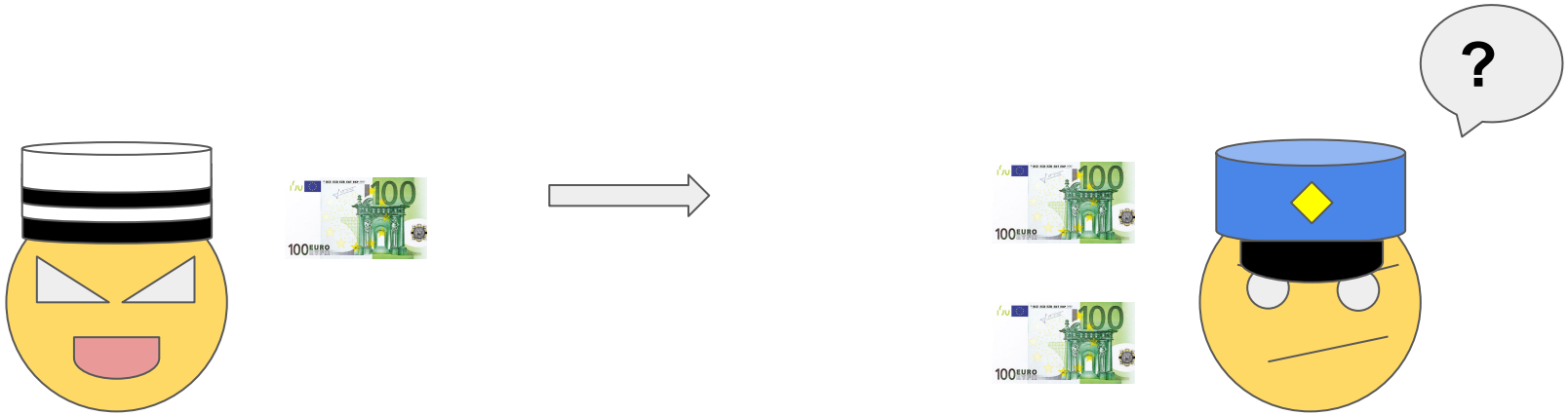
Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.



Adversarial Training analogy

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.

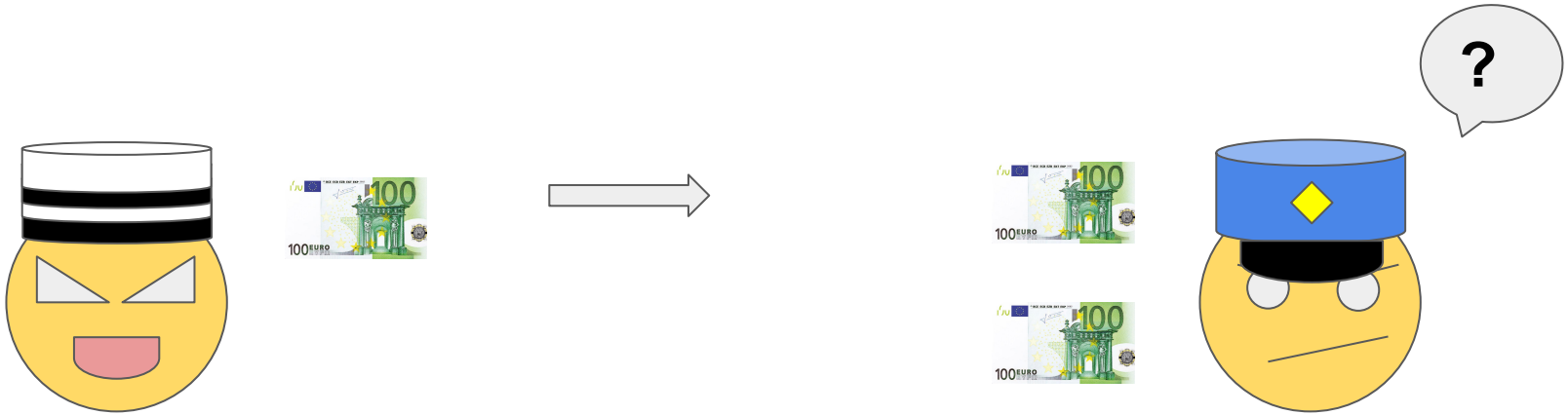
After enough iterations, and if the counterfeiter is good enough (in terms of **G** network it means “has enough parameters”), the police should be confused.



Adversarial Training analogy

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.

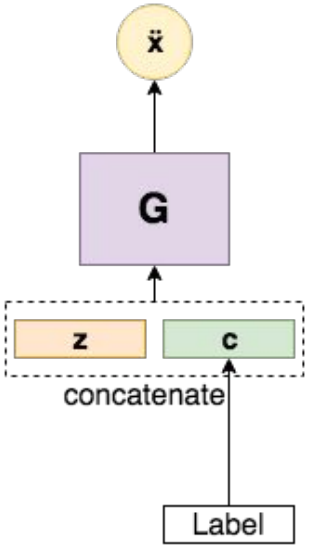
After enough iterations, and if the counterfeiter is good enough (in terms of **G** network it means “has enough parameters”), the police should be confused.



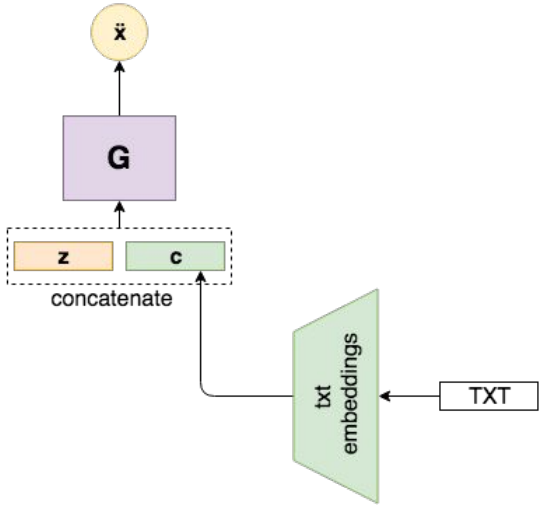
Conditioned GANs

GANs can be conditioned on other info extra to **z**: text, labels, etc..

z might capture random characteristics of the data, variabilities of possible futures, whilst **c** would condition the deterministic parts



For details on ways to condition GANs:
[Ways of Conditioning Generative Adversarial Networks (Wack et al.)]



GAN Applications

So far GANs have been extensively used in computer vision tasks:

- Generating images/generating video frames
- Unsupervised feature extraction/learning representations
- Manipulating images (in a photoshop advanced level)
- Image coding/Super Resolution
- Transferring image styles

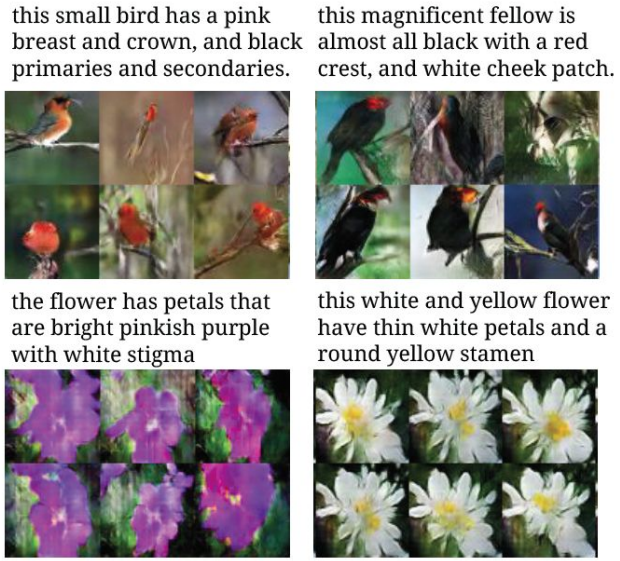
However we have been working on advances for speech generation!

Generating images/frames

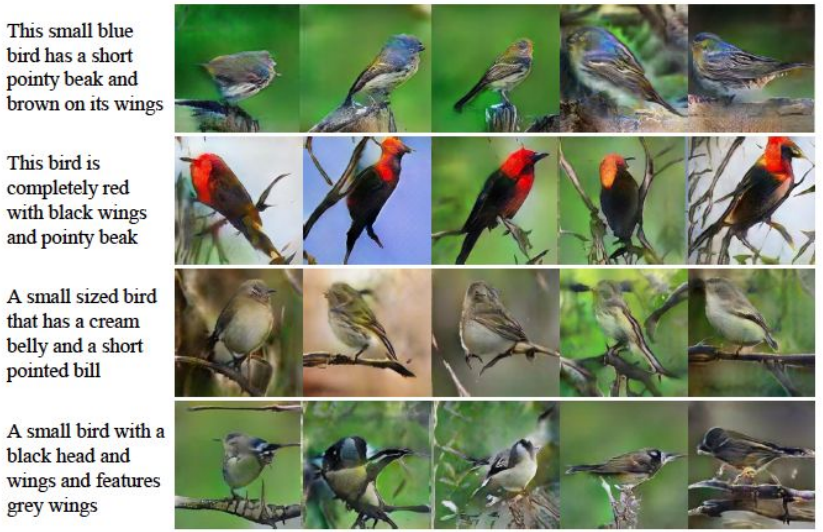
Deep Conv. GAN (DCGAN) effectively generated 64x64 RGB images in a single shot. For example bedrooms from LSUN dataset.

(Radford et al. 2015)

Generating images/frames conditioned on captions



(Reed et al. 2016b)

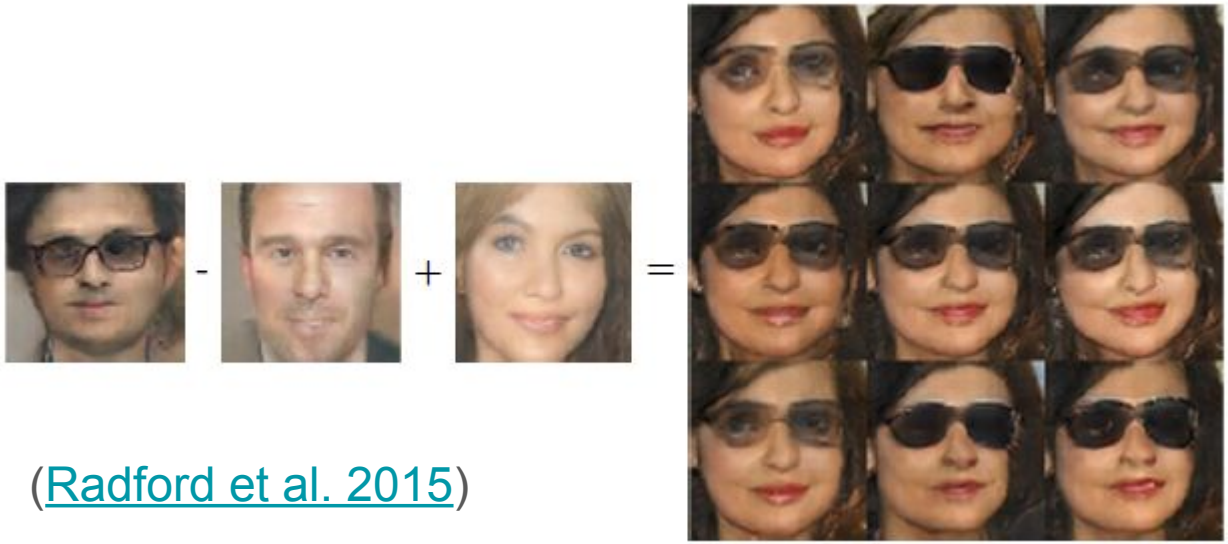


(Zhang et al. 2016)

Unsupervised feature extraction/learning representations

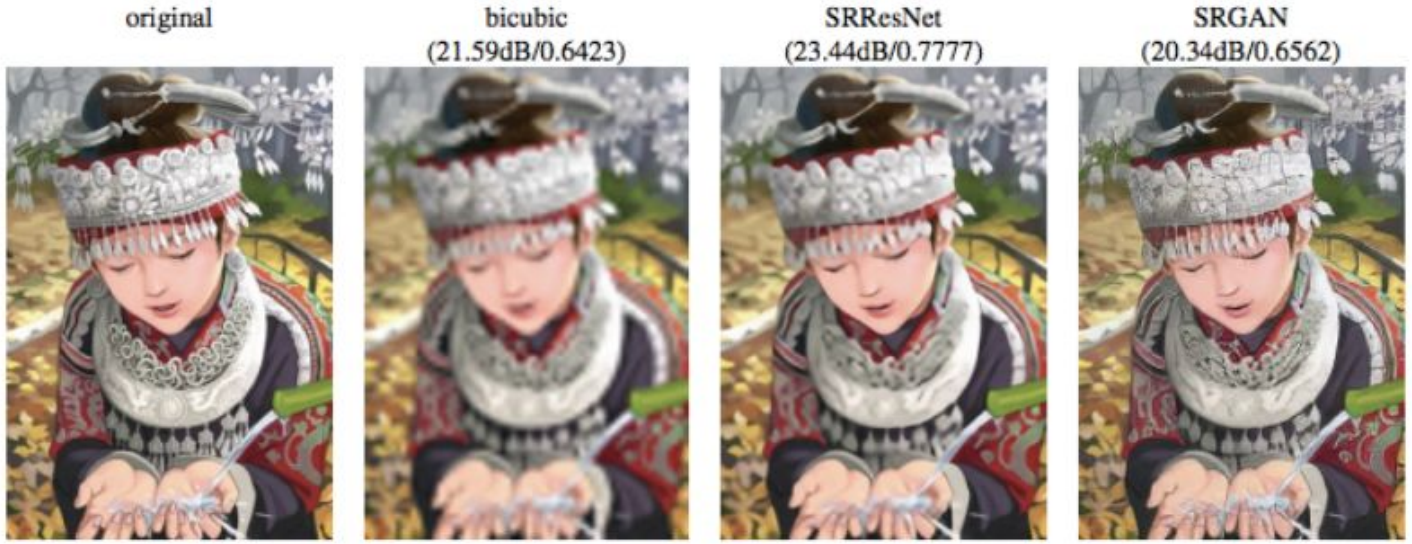
Similarly to word2vec, GANs learn a distributed representation that disentangles concepts such that we can perform operations on the data manifold:

v(Man with glasses) - v(man) + v(woman) = v(woman with glasses)



(Radford et al. 2015)

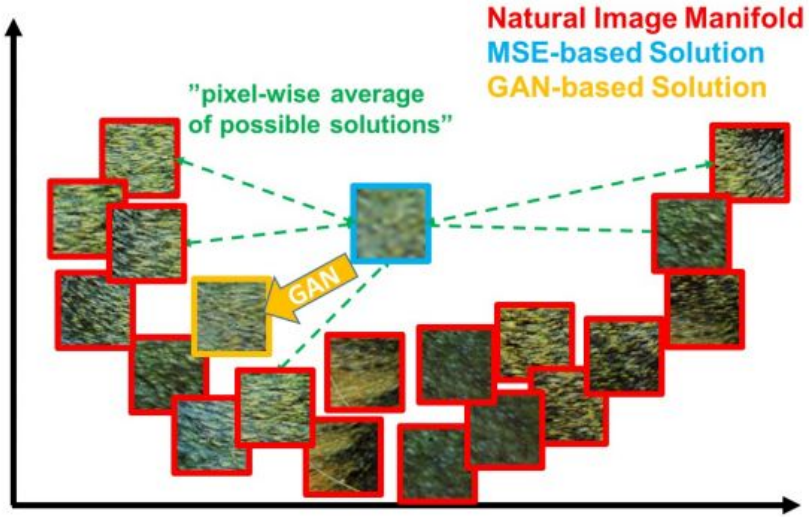
Image super-resolution



(Ledig et al. 2016)

Bicubic: not using data statistics. SRResNet: trained with MSE. SRGAN is able to understand that there are multiple correct answers, rather than averaging.

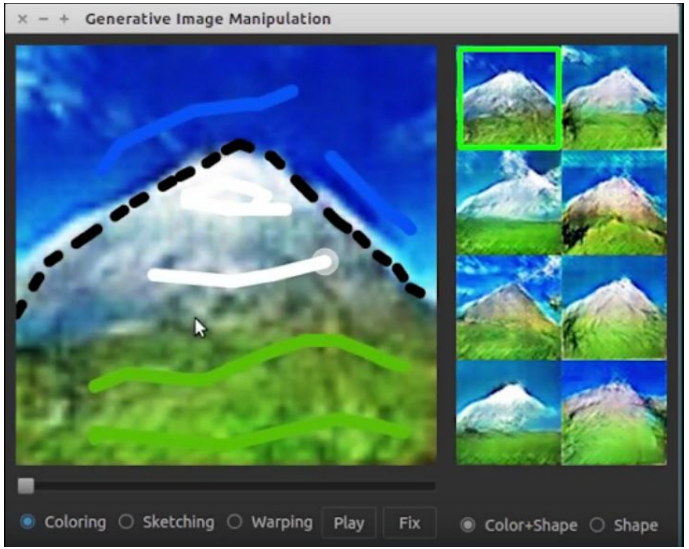
Image super-resolution



(Ledig et al. 2016)

Averaging is a serious problem we face when dealing with complex distributions.

Manipulating images and assisted content creation

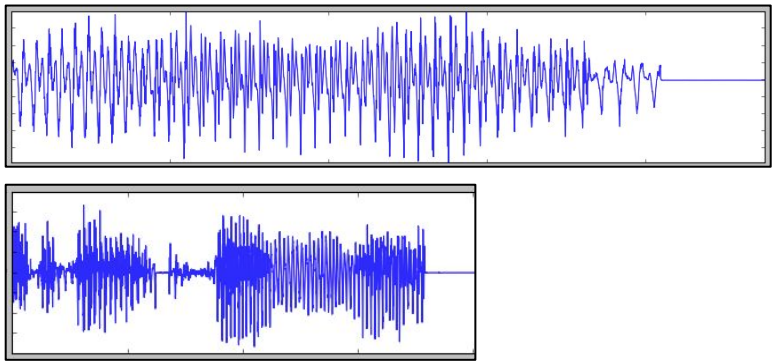


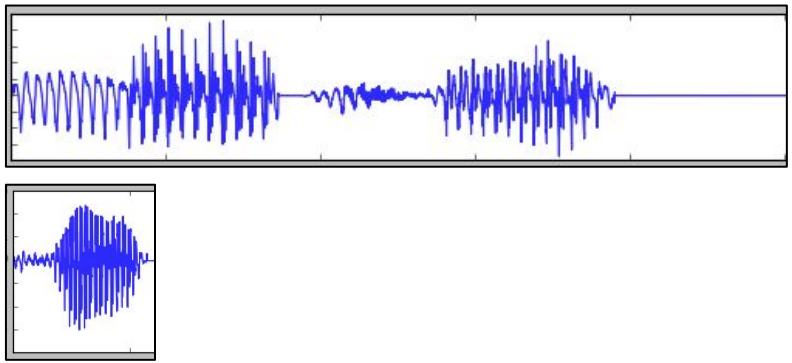
https://youtu.be/9c4z6YsBGQ0?t=126 https://youtu.be/9c4z6YsBGQ0?t=161

(Zhu et al. 2016)

Waveforms generation

We have done recent advances in generating speech signals with GAN models. There are none existent systems doing so until now. Current line of research.





(S.Pascual, A.Bonafonte, J.Serrà)

Caveats

Where is the downside...?

Well GANs are tricky and hard to train! We do not want to minimize a cost function. Instead we want both networks to reach a Nash equilibria (saddle point).

Because of extensive experience within the GAN community (with some does-not-work-frustration from time to time), you can find some tricks and tips on how to train a GAN here: https://github.com/soumith/ganhacks

An open problem of GANs is the evaluation of their generation quality: there are no established objective metrics to do so → We look (or hear) the generated samples to know when to stop training or how are we doing.

Thanks ! Q&A ?

Strongly recommended reading: NIPS 2016 Tutorial: Generative Adversarial Networks (I. Goodfellow)

@Santty128