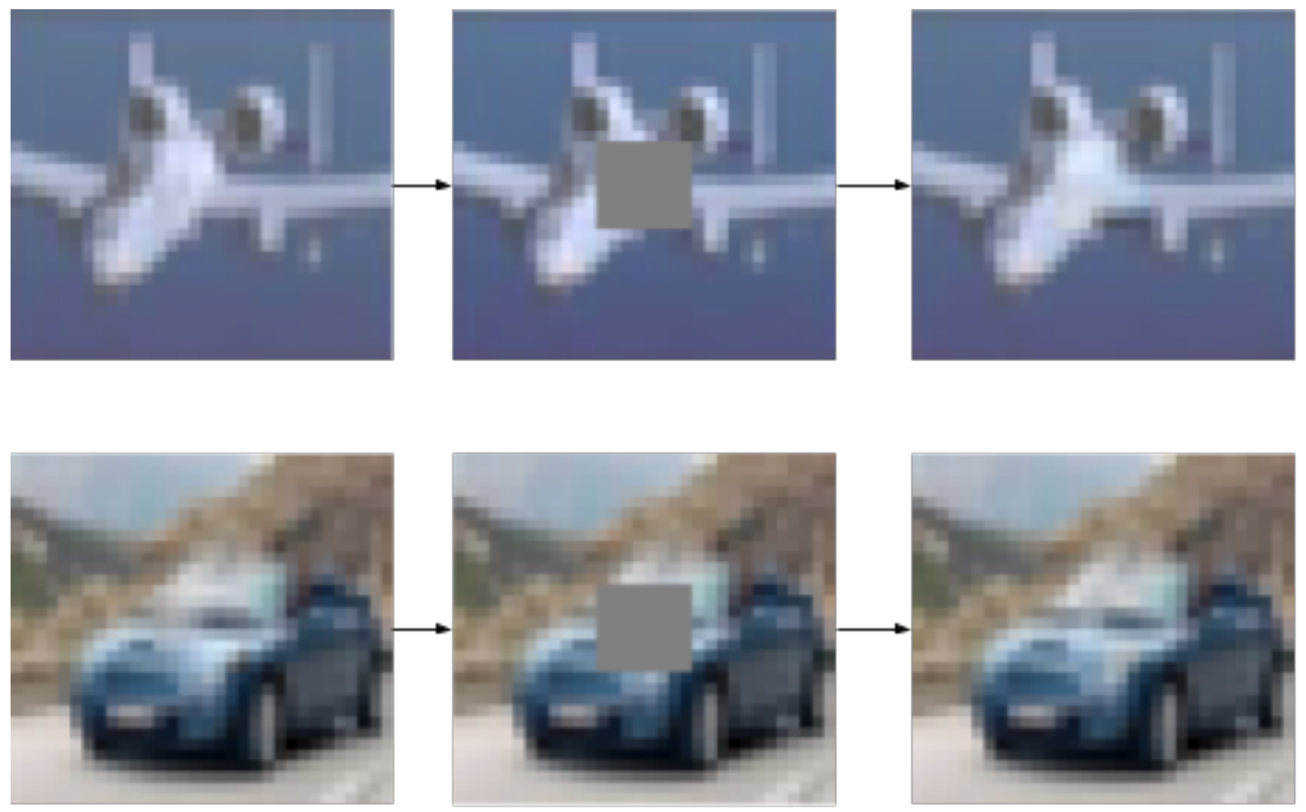
Deep Image Inpainting

Carlito Burlaco, L. Duperier, Y. Le Calonnec,

Problem



Original Images — Input Images — Output Images

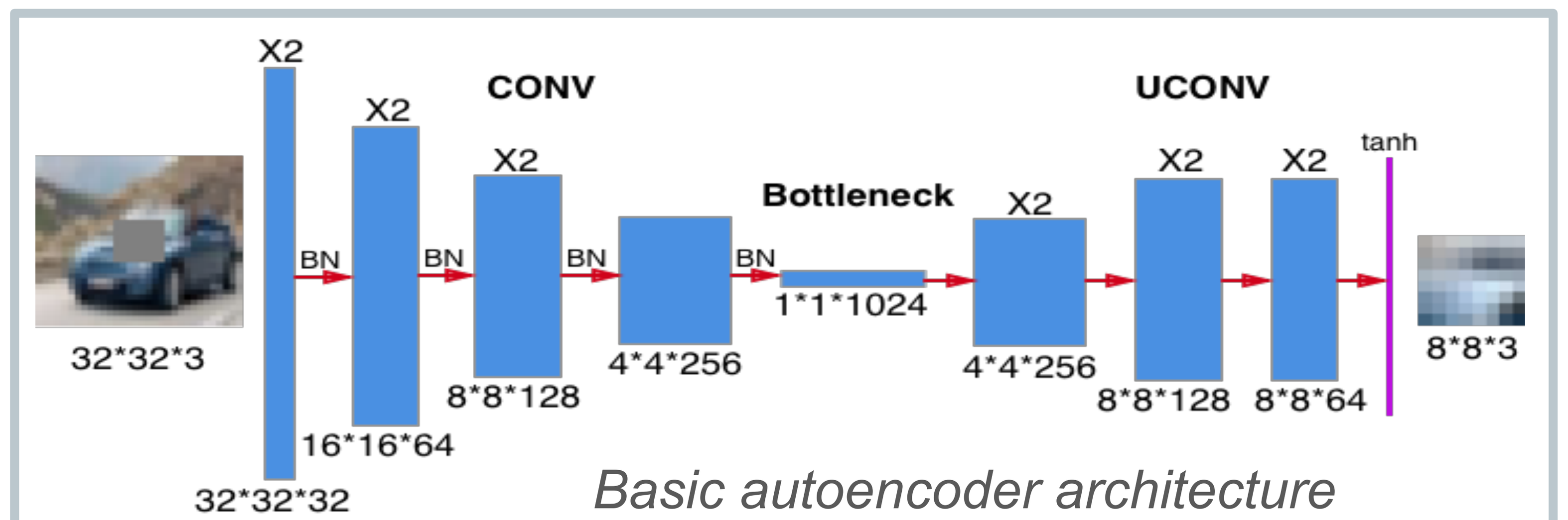
- **Reconstruct** missing parts of images
- Crucial for: **restoration**, image **editing**
- Previous approaches:
  - Repeat textures/patterns but can't reconstruct objects
  - No real understanding of context
- Evaluation: pixel-wise L2 loss
  Real: $Y \in \mathbb{R}^{n \times n \times 3}$ vs prediction: $\hat{Y} \in \mathbb{R}^{n \times n \times 3}$

$$L_i = \sum_{p,q,r} (Y^{(i)}_{p,q,r} - \hat{Y}^{(i)}_{p,q,r})^2$$

Dataset

- **CIFAR10**: 50k train / 10k tests
- 80m tiny images (32x32x3, 8x8 center missing)
- **Data augmentation**: random hue, saturation, blur...

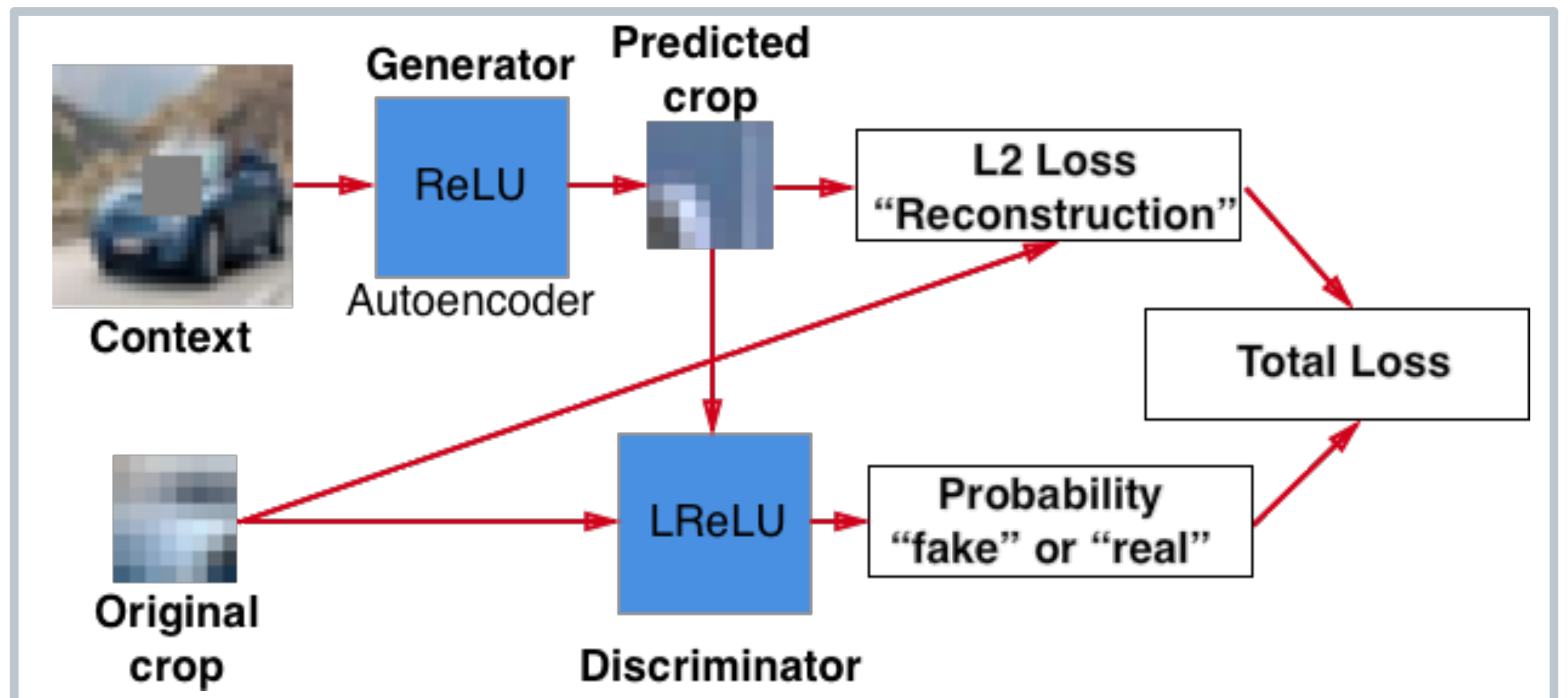
Autoencoder



Basic autoencoder architecture

DC-GAN

- **Bottleneck** forces network to learn a **high-level**, compact representation of the context.
- We tried different **architectures** for encoder: VGG, Inception, ResNets, AlexNet.
- **Transfer learning**: use pre-trained first layers for encoder.



High-level overview of autoencoder architecture

- **Discriminator** (D) is a CNN trained to **discern** real from fake images.
- **Generator** (G) is an autoencoder.
- Usually in GANs, G generates outputs from random inputs: we use the **context** instead.
- G improves to fool D, and D learns from G's techniques: extremely **hard to train** in practice.
- Train both networks in **parallel**.
- **No FC** for deeper architectures.

W-GAN

- D outputs an unbounded score, not a probability.
- Train D to convergence at each iteration.
- Clip gradients to ensure Lipschitzianity.

Held out example from our best architecture

Pixel CNN & RNN

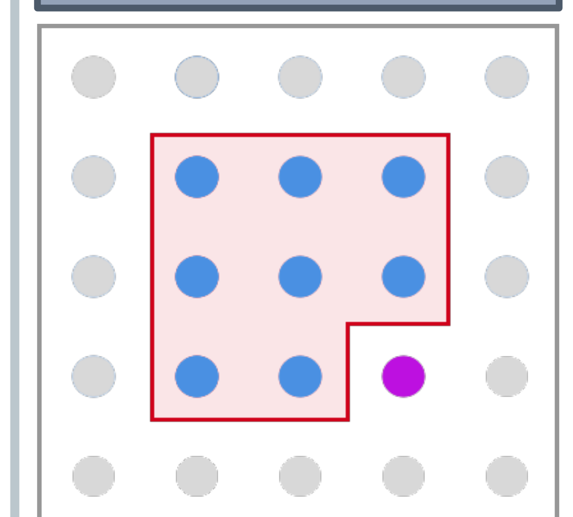
- Image distribution with **conditional** pixel probabilities.

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

- Split along channels.

$$p(x_{i,R}|\mathbf{x}_{<i})p(x_{i,G}|\mathbf{x}_{<i}, x_{i,R})p(x_{i,B}|\mathbf{x}_{<i}, x_{i,R}, x_{i,G})$$

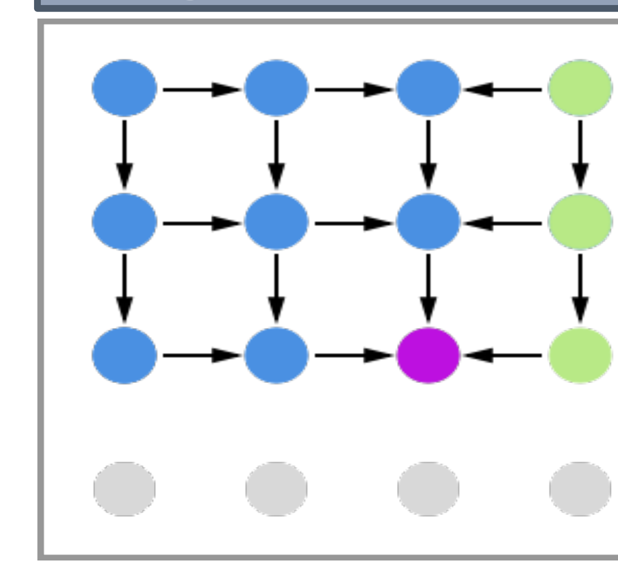
Pixel CNN



- Use **local information** to **recursively** infer pixel distribution.
- Apply a **convolution** layer on the selected (red) zone to get a 3x256-dimensional vector.
- Parameters **shared** across pixels.
- **Softmax** layer gives probabilities for each channel.

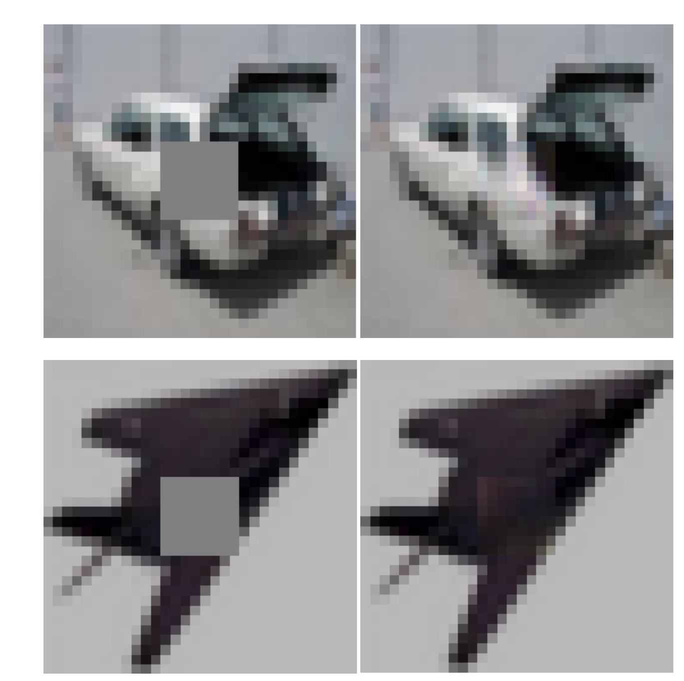
Diagonal BiLSTM

- Use information from **top rows**.
- LSTM top-left → bottom-right
- LSTM top-right → bottom-left
- **Mix distributions** from the two LSTMs to get the distribution of a given pixel.
- Predict channels in a specific order: R → G → B



Results / Conclusion

Model	L2 Loss
Euclidian CNN	8.56
DCGAN	7.49
WGAN	**4.26**
Pixel CNN	6.98
Diagonal BiLSTM	5.27



- WGAN + deep autoencoder: highest performance
- It is crucial to deeply understand the occluded objects
- Future work: better GAN training, larger images