Timişoara Machine Learning Workshop, Feb 23, 2019

ConvNets Tutorial

Viorica Patraucean DeepMind

- Download practical_session_colorisation from gdrive
- Unzip to /my/path
- Go to https://colab.research.google.com
- Upload /my/path/Colorisation_start.ipynb
- Runtime -> Change runtime type -> Hardware accelerator: GPU
- Connect

Common training paradigms

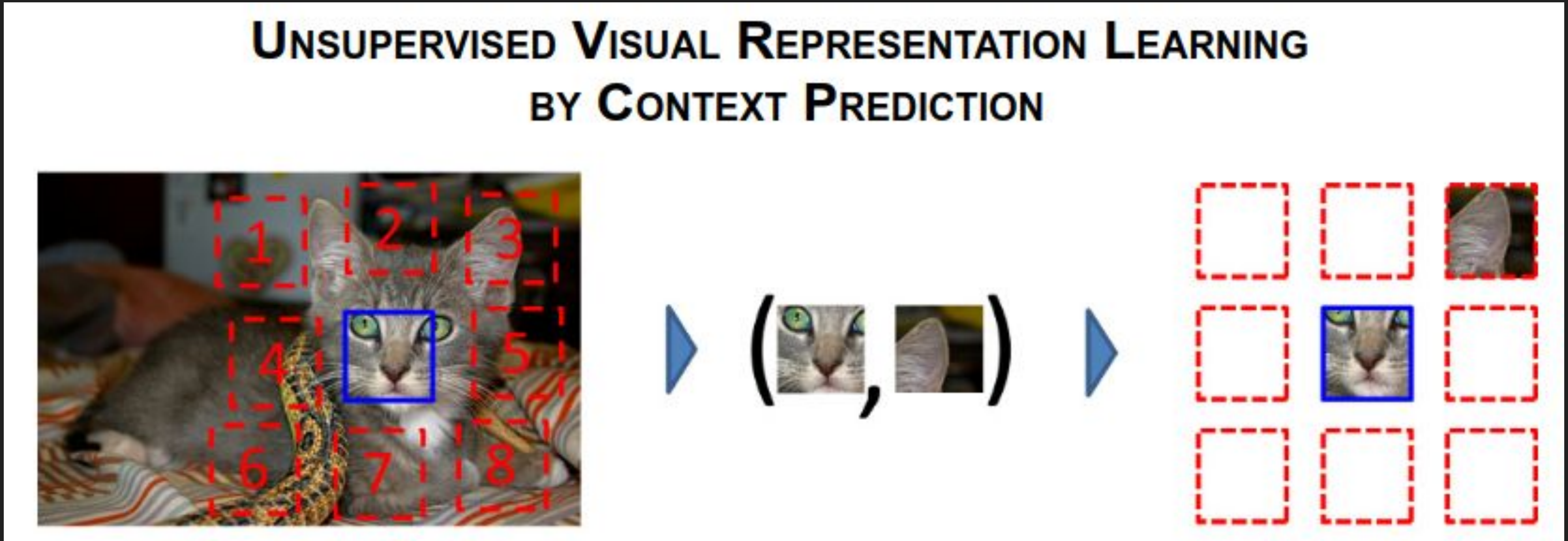
Supervised learning: model conditional distribution p(y|x), y=labels provided by (human) experts; e.g. image classification, machine translation, etc.

Unsupervised learning: model the distribution of the input data p(x), e.g. clustering, autoencoders, GANs etc.

Reinforcement learning: learn optimal policy, maximise reward

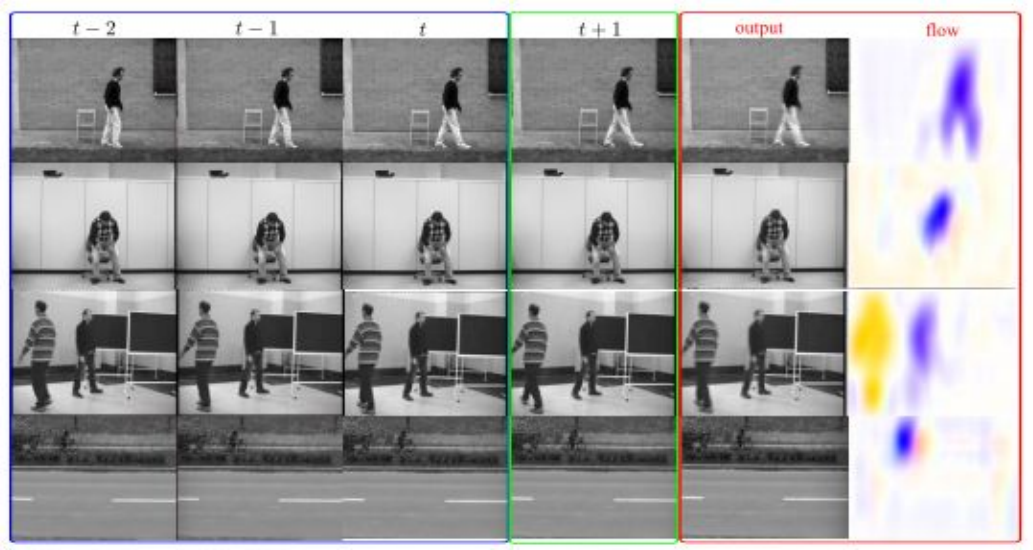
Self-supervised learning: supervised, but labels do not require human expert

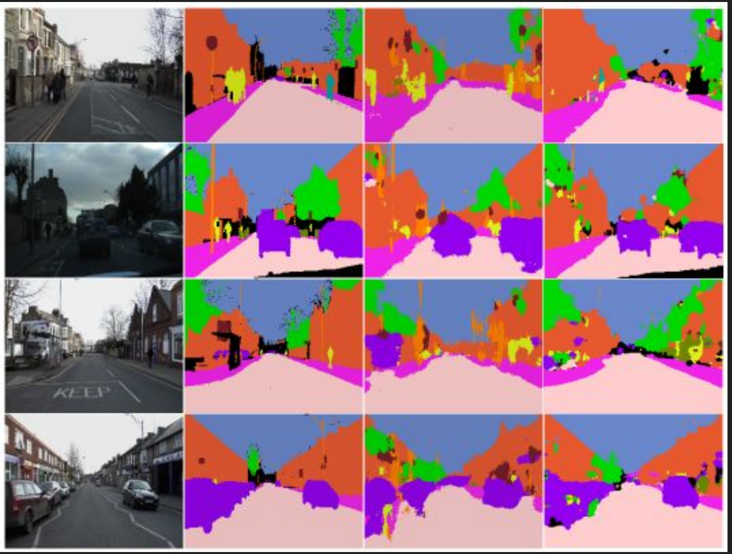
Examples of self-supervised learning tasks



Doersch et al., ICCV2015

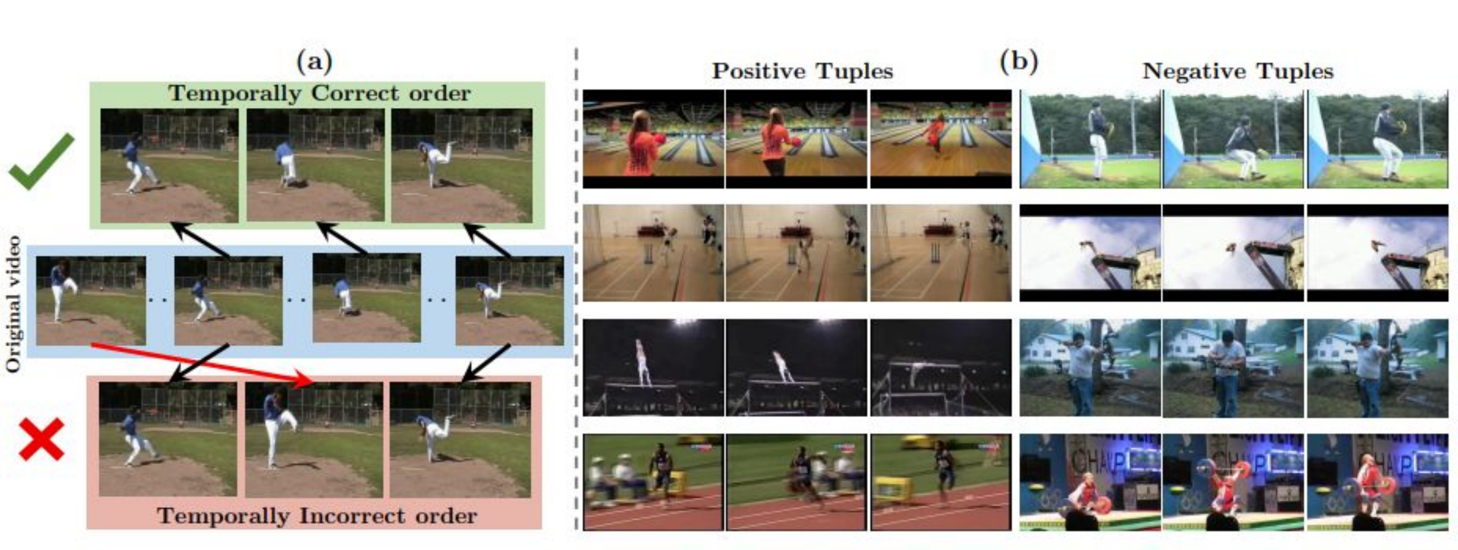
Examples of self-supervised learning tasks





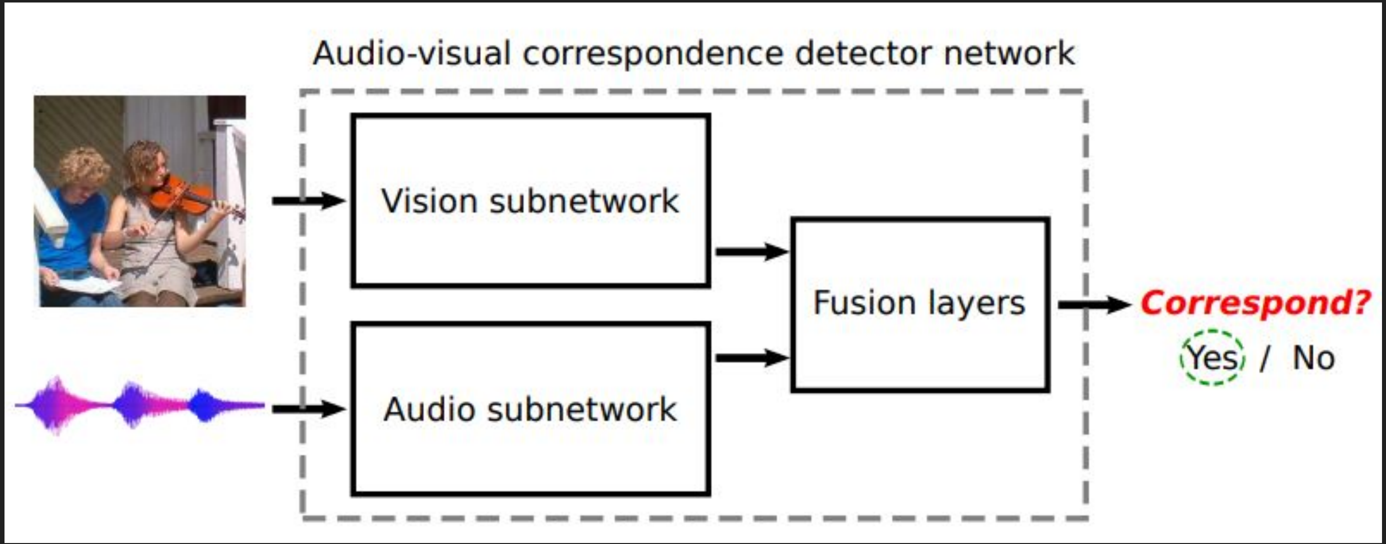
Spatio-temporal video autoencoder with differentiable memory, Patraucean et al., ICLR 2016

Examples of self-supervised learning tasks



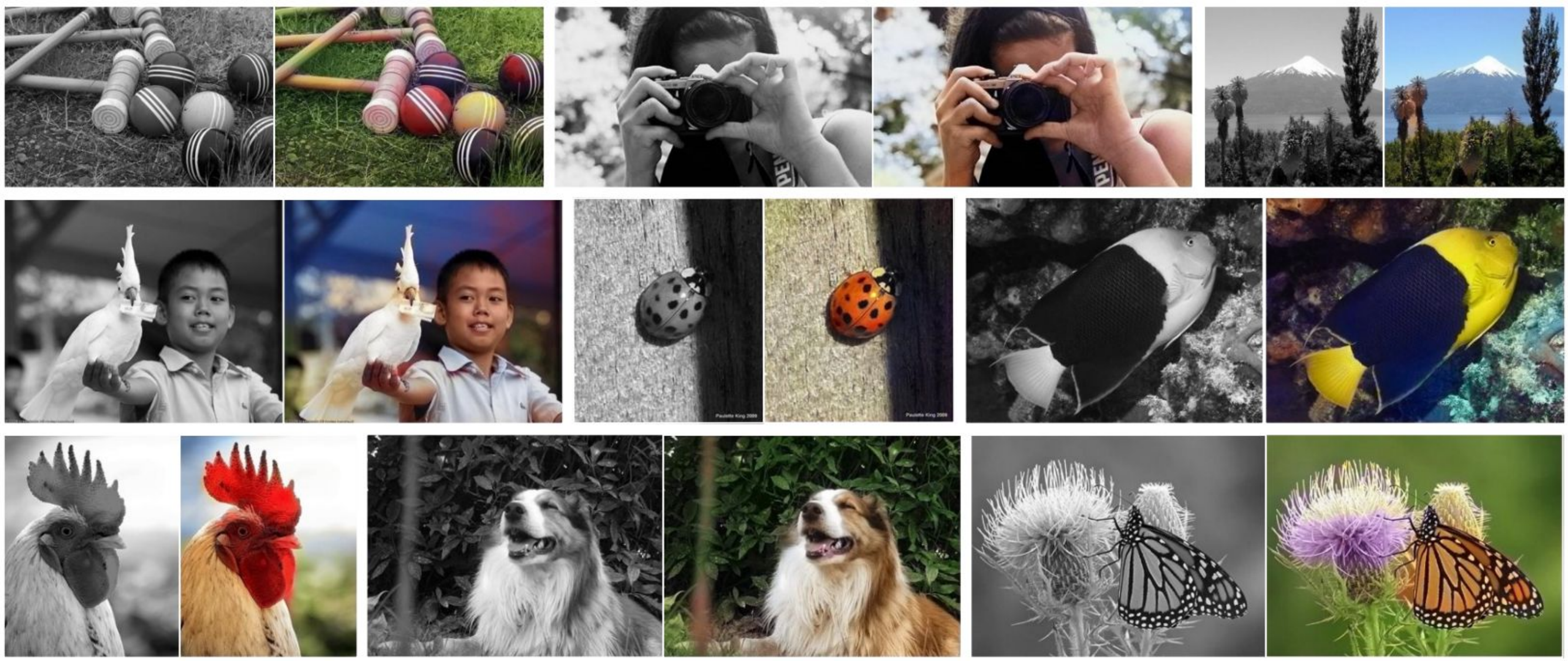
Shuffle and Learn: Unsupervised Learning using Temporal Order Verification, Misra et al., ECCV2016

Examples of self-supervised learning tasks



Look, Listen and Learn, Arandjelovic and Zisserman, ICCV2017

Colorful image colorization, Richard Zhang, Phillip Isola, Alexei A. Efros, ECCV16



Colour and semantics

Setting up a (self-)supervised learning task

1. Define a task, structure of inputs and outputs
2. Define an objective (loss) function
3. Define an architecture
4. Define an optimiser to minimise the loss
5. Get dataset (training / validation / test): input data, ground truth
6. Train the network on training set
7. Evaluate on validation set for hyperparameter tuning
8. Diagnose behaviour (e.g. overfitting) and iterate
9. Evaluate on unseen test set

Define task: Image colourisation

Let $I = (L, a, b)$ be a colour image with (H, W) pixels $I \in \mathbb{R}^{HxWx3}$

Original image I

Colour spaces: RGB

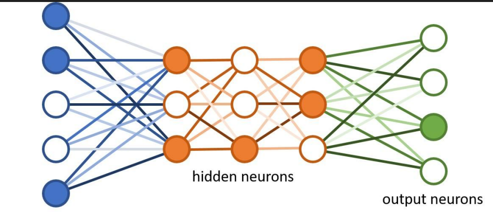
Colour spaces: Lab

Lightness [0, 100]　　Green-Red [-110, 110]　　Blue-Yellow [-110, 110]

Define task: Image colourisation

Given the **input** lightness channel $L \in \mathbb{R}^{HxWx1}$, learn a mapping $\hat{y} = f(L)$ to the other two channels $y = (a, b) \in \mathbb{R}^{HxWx2}$, $\hat{y}$ = **output** - dense prediction

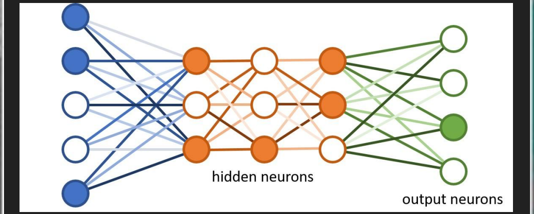
Input lightness channel *L*



Output *ab* channels

Define loss

Dense prediction problem: pixel-wise loss



Input lightness channel *L*

Output *ab* channels

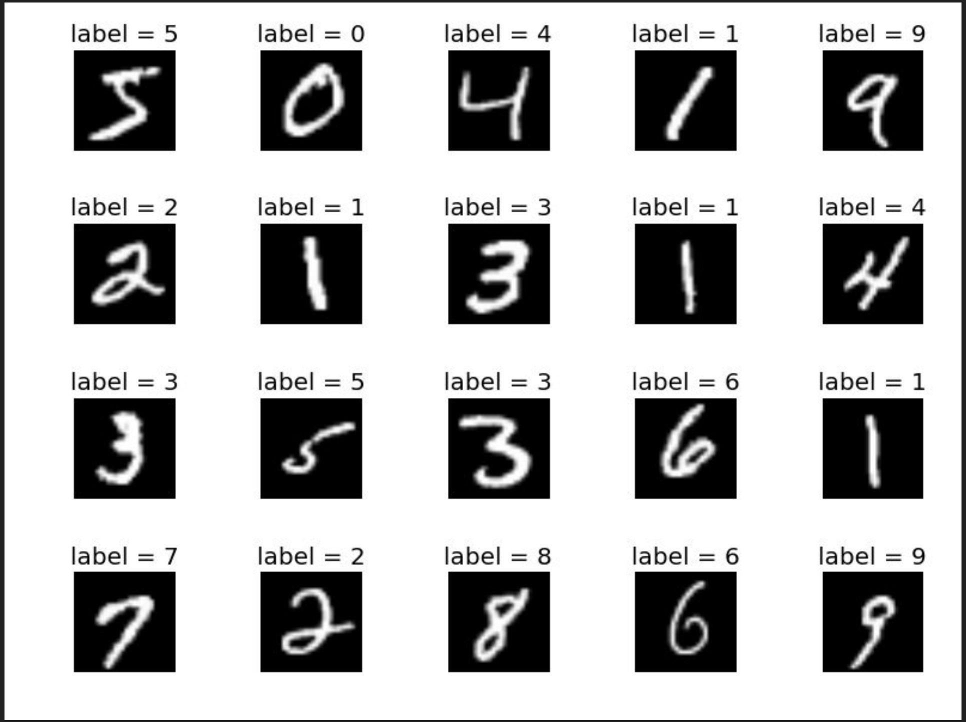
Define loss

Pixel-wise loss (similar to e.g. semantic segmentation)

Cityscapes dataset

Define loss

Loss: **regression** vs **classification**



MNIST digit recognition:

- **Regression**: output a real number as close as possible to the label
  E.g. label=5, pred=4.99 or 5.001
  Loss: l2_loss $L(y, \hat{y}) = \frac{1}{2}||y - \hat{y}||_2^2$

- **Classification**: output a probability distribution over the possible classes:
  E.g. **label**=5,
  Ground truth 0 0 0 0 0 1 0 0 0 0
  (one-hot; dirac)

  Pred: 0 0 0.15 0 0 0.8 0 0 0 0.05

  Loss: softmax_cross_entropy

$$L(p, q) = -\sum_{x \in X} p(x) \log q(x)$$

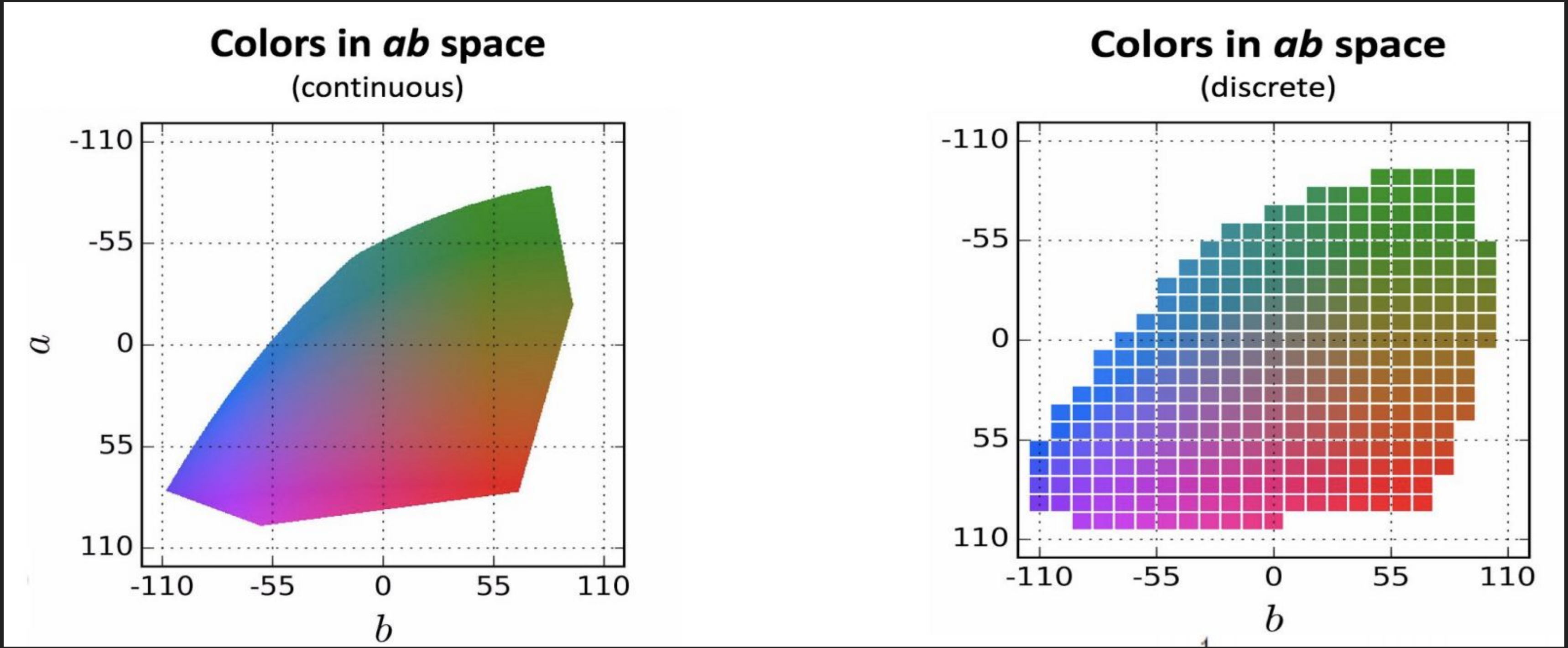
Define loss

1st option: **pixel-wise regression** $\mathbf{L}_2(y, \hat{y}) = \frac{1}{2}\sum_{h,w}^{H,W} ||y_{h,w} - \hat{y}_{h,w}||_2^2$

Issue: multimodal problem; would converge to the mean = gray-ish values

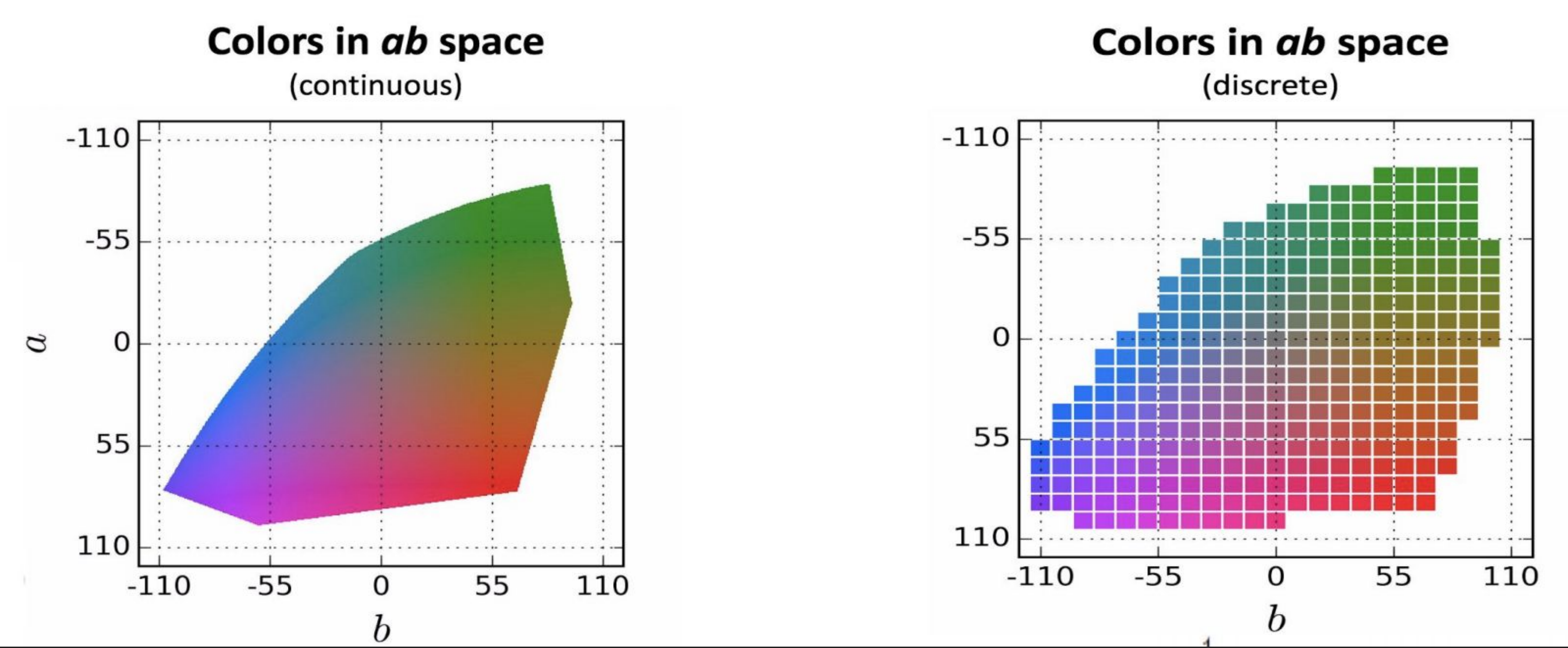
Define loss

Classification; discretise colour space; keep *N* bins. N=313



Define loss

Learn mapping to a probability distribution over possible bins



Define loss

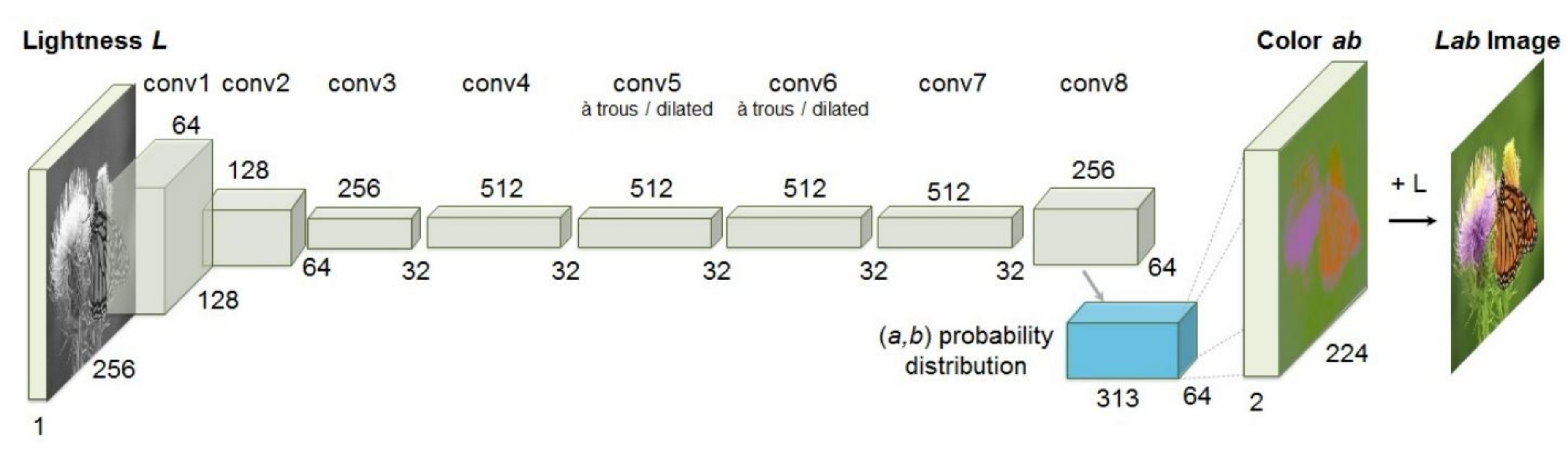
Learn mapping to a probability distribution over possible bins

Output $\hat{y} \in [0, 1]^{H \times W \times N}$

Pixel-wise cross entropy loss: $L(y, \hat{y}) = -\sum_{h,w}^{H,W} \sum_{i}^{N} y_{h,w,i} \log \hat{y}_{h,w,i}$

Define architecture

Dense output (not a single label as in classification).

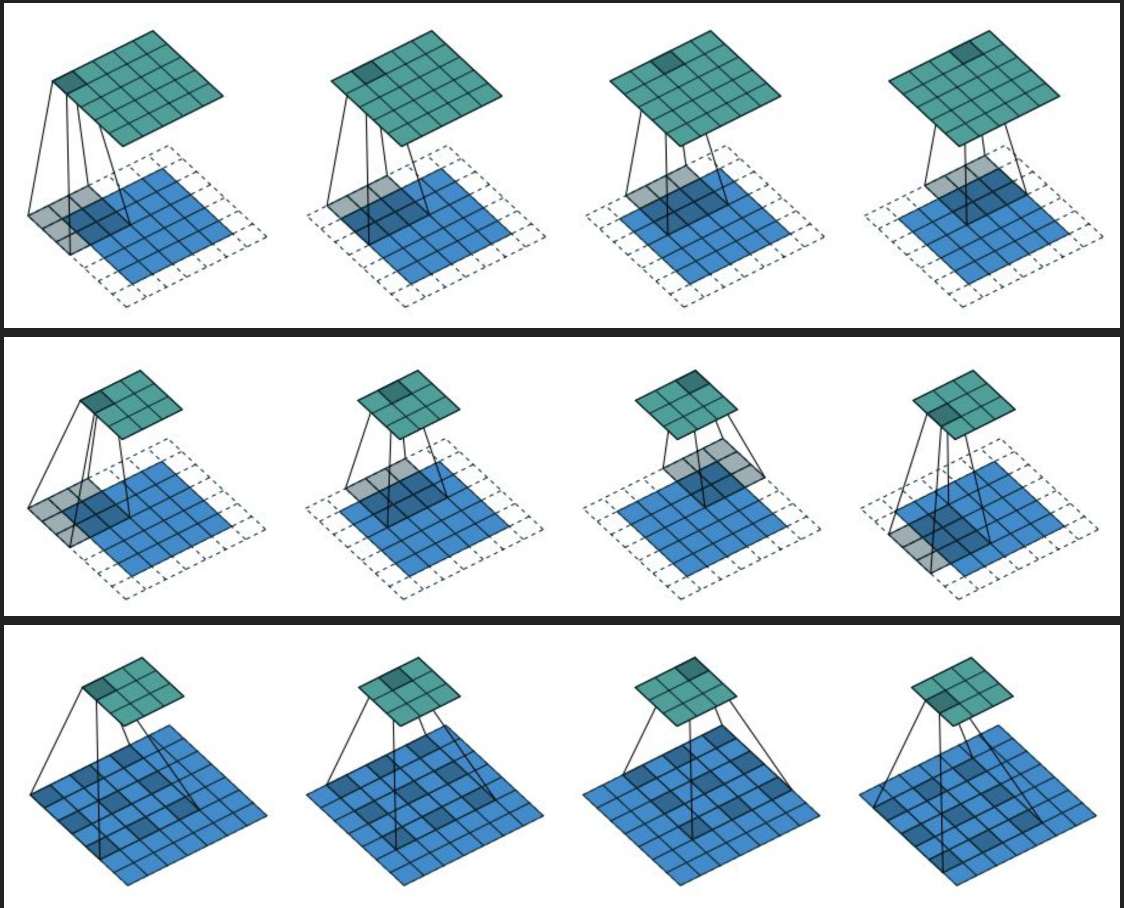


VGG architecture

conv# = (conv+ReLU) x2 or x3, followed by BatchNorm layer

no pooling layers; down(up)sampling done through dilated convs / deconvs

Conv Ops

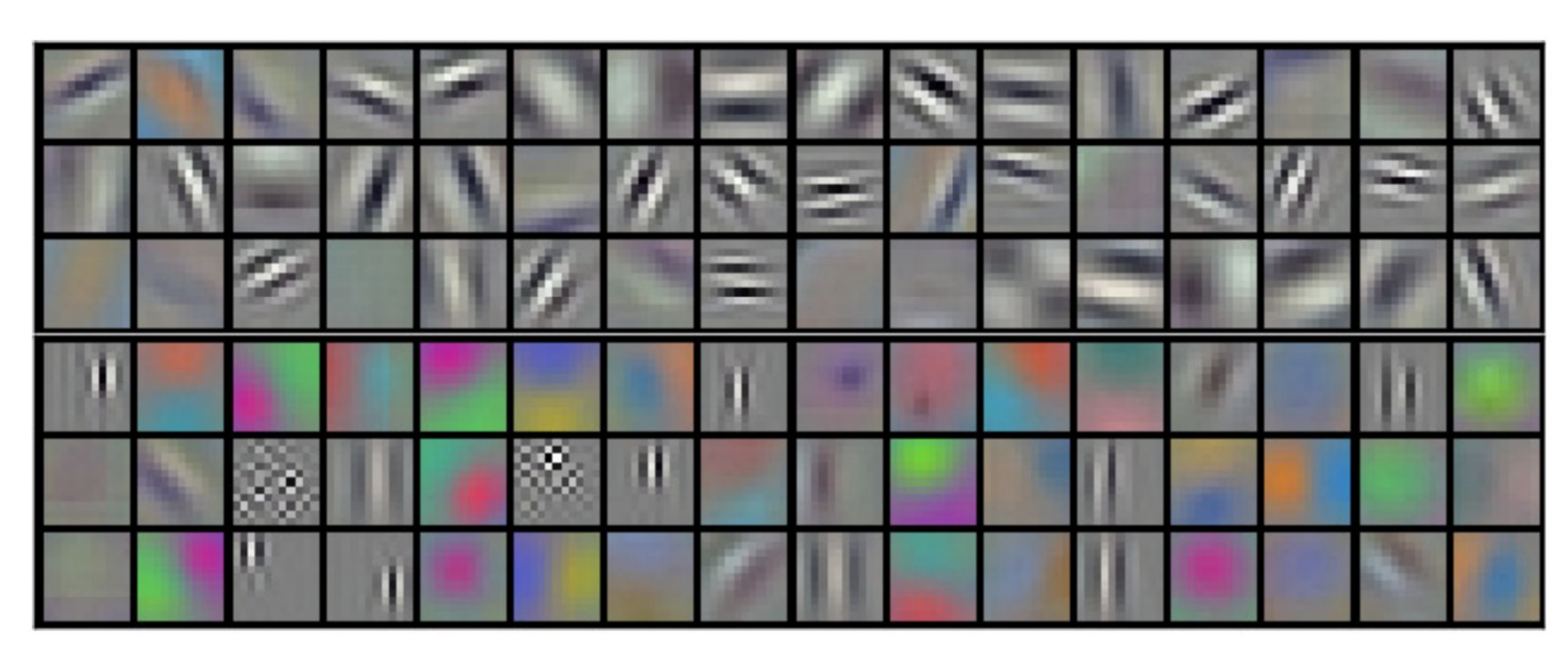


Input 5x5
Kernel 3x3
Stride 1
Rate (dilation) 1
Padding SAME Output 5x5
Padding VALID Output 3x3

Stride 2

Rate 2

First layer learnt filters: contours



Define optimiser

Some version of stochastic gradient descent

- Training on mini-batches
- Vanilla SGD
- SGD with momentum
- Adam (second order approximation)
- etc.

Dataset

Any image dataset works (e.g. Imagenet, Cifar10, etc.)

We use Tiny Imagenet (subset of Imagenet):

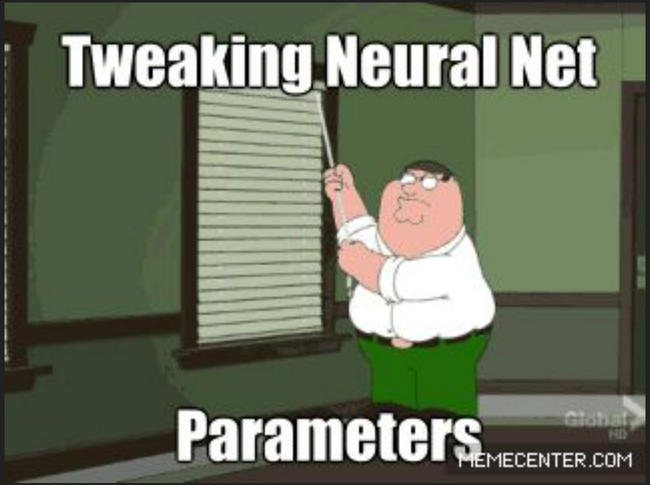
- 200 classes
- 500 training images per class; 50 validation, 50 test; 64x64 pixels

(a) Tiny ImageNet samples.

Hyperparameter tuning

Types of parameters in a neural network:

- Trainable (e.g. through SGD)
- Hyperparameters (non-trainable, hand-tuned)



Architecture hyperparameters:

- Number of layers
- Number of features per layer
- Filter sizes, strides

Training hyperparameters:

- Learning rate
- Learning rate decay
- Momentum, etc.
- Batch size

Alternative: evolutionary strategies, e.g. Population-based training, Jaderberg et al, 2017

Input

Original

Output

Input

Original

Output

Input

Original

Output

Input

Original

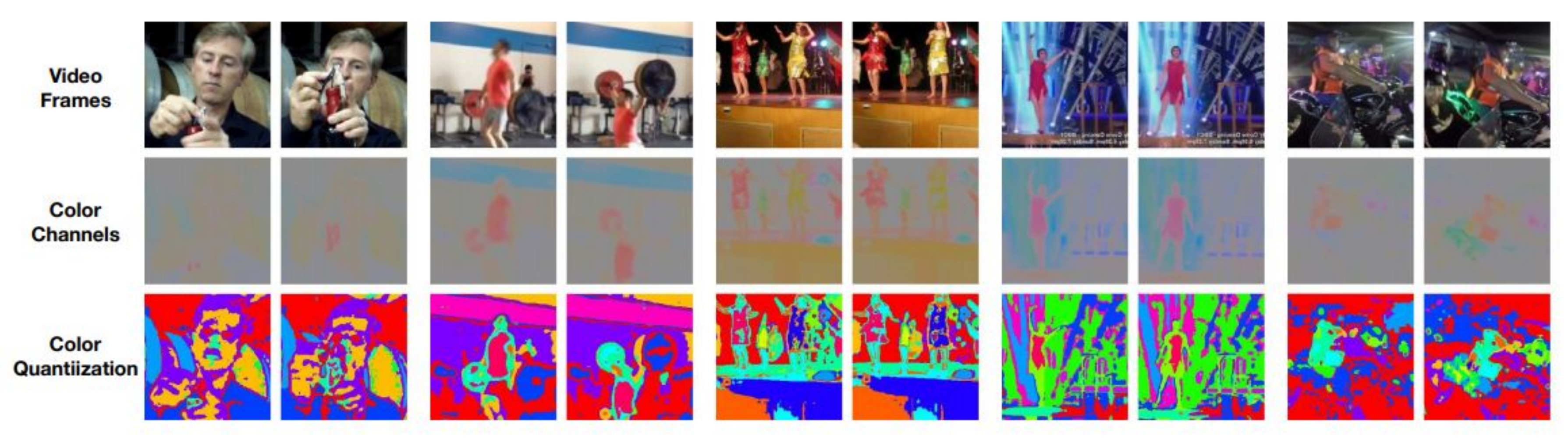
Output


Input


Output


Original

Extension to video



Tracking emerges by colorizing videos, Vondrick et al, ECCV2018