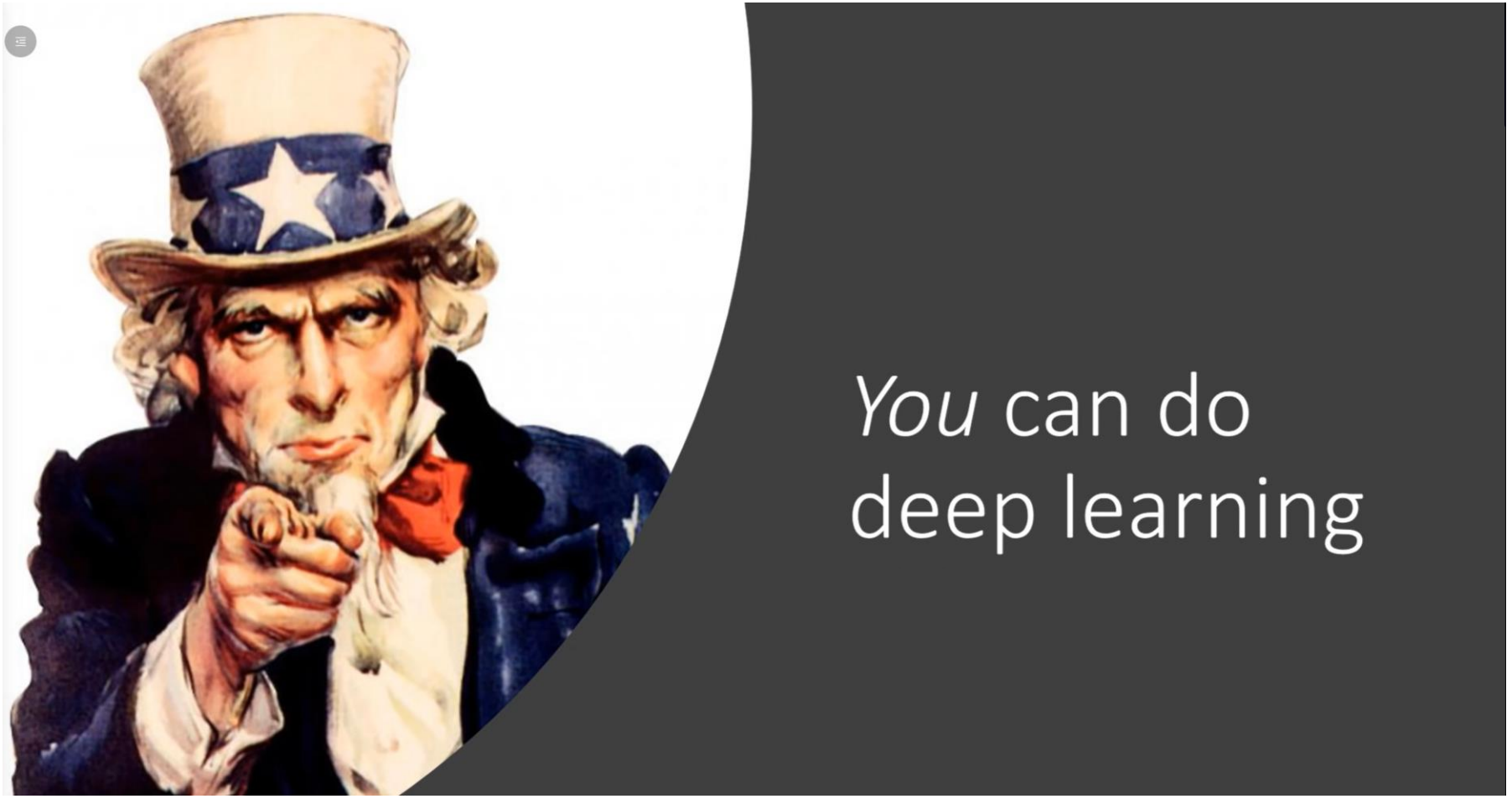
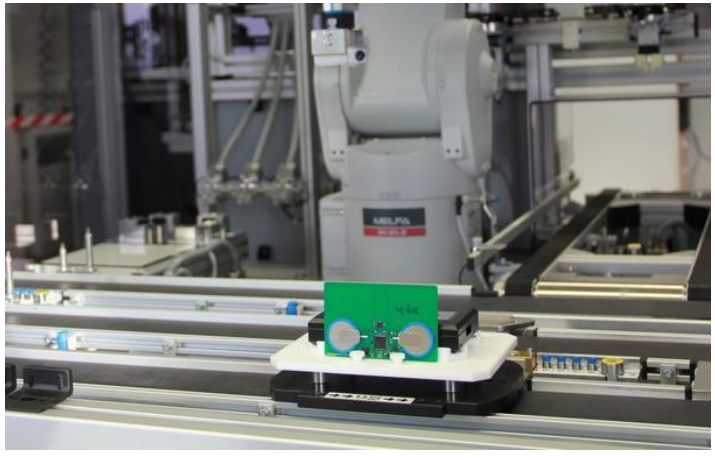
Anomaly Detection with Deep Learning



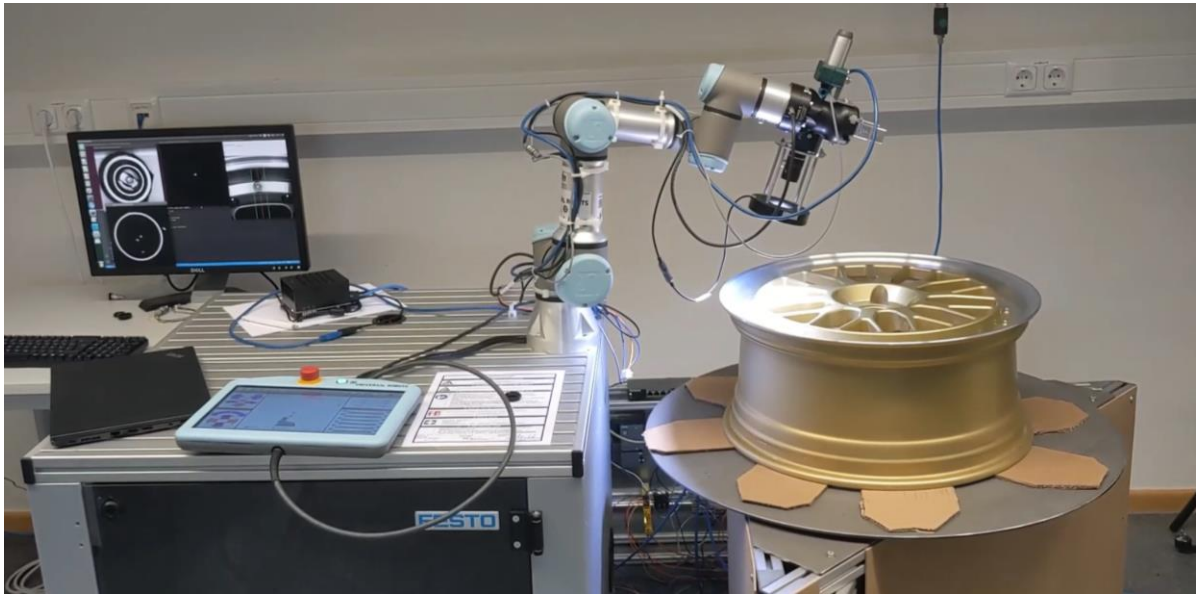
Where I am.

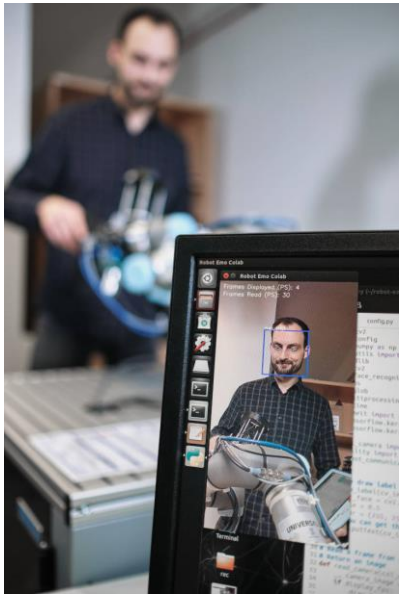
HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN
UNIVERSITY OF APPLIED SCIENCES

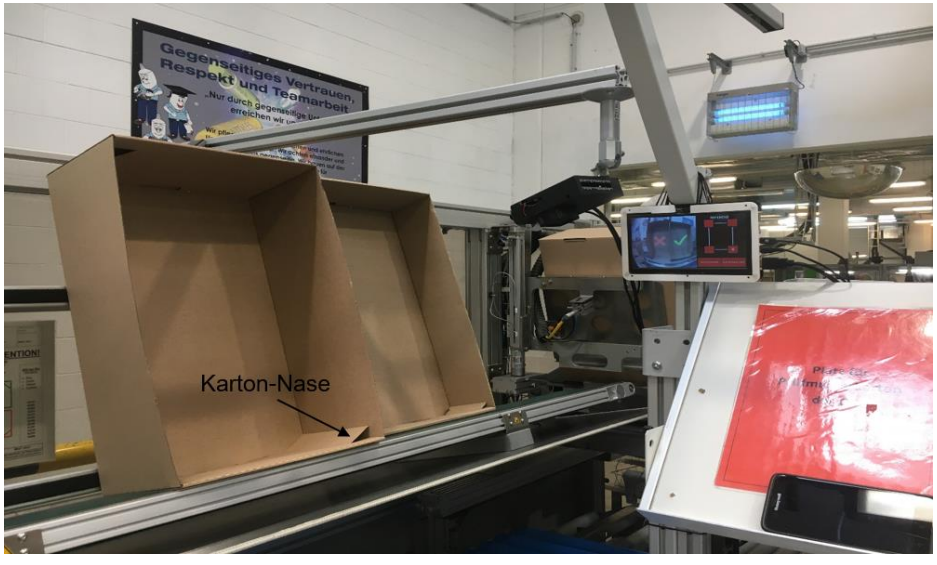
- Johannes Metzler, research associate in research group **"Smart Production System“** from Prof. Dirk Reichelt
- Expert in **Computer Vision** and **Artificial Intelligence**

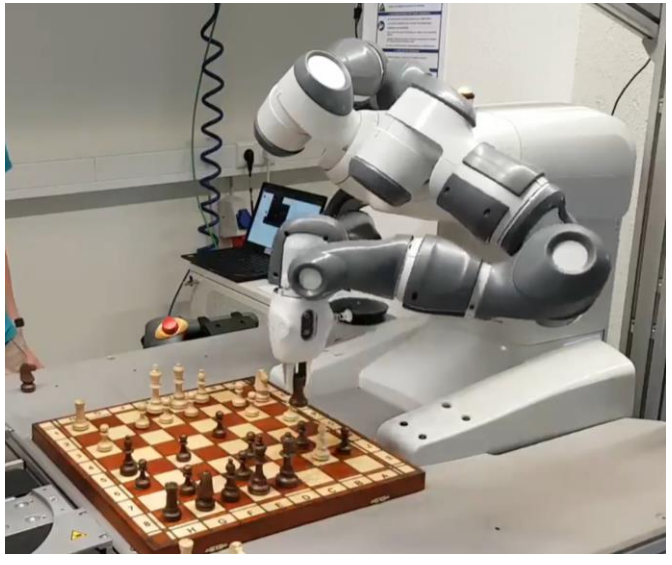


Projects done.

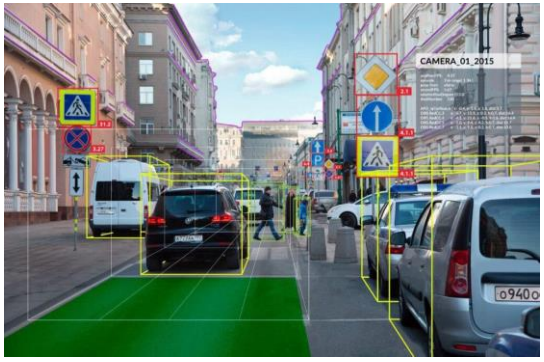




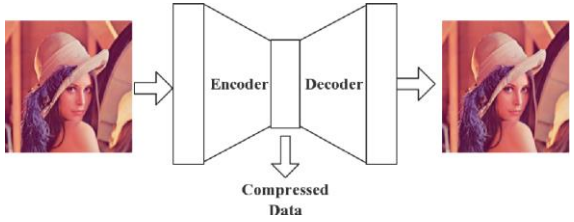




What we do in this course?

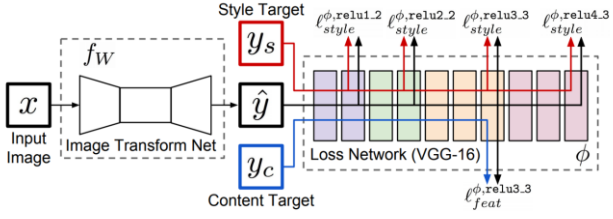


Foundations of Computer Vision



Anomaly Detection with Autoencoders

Boosting Autoencoders with Feature Loss



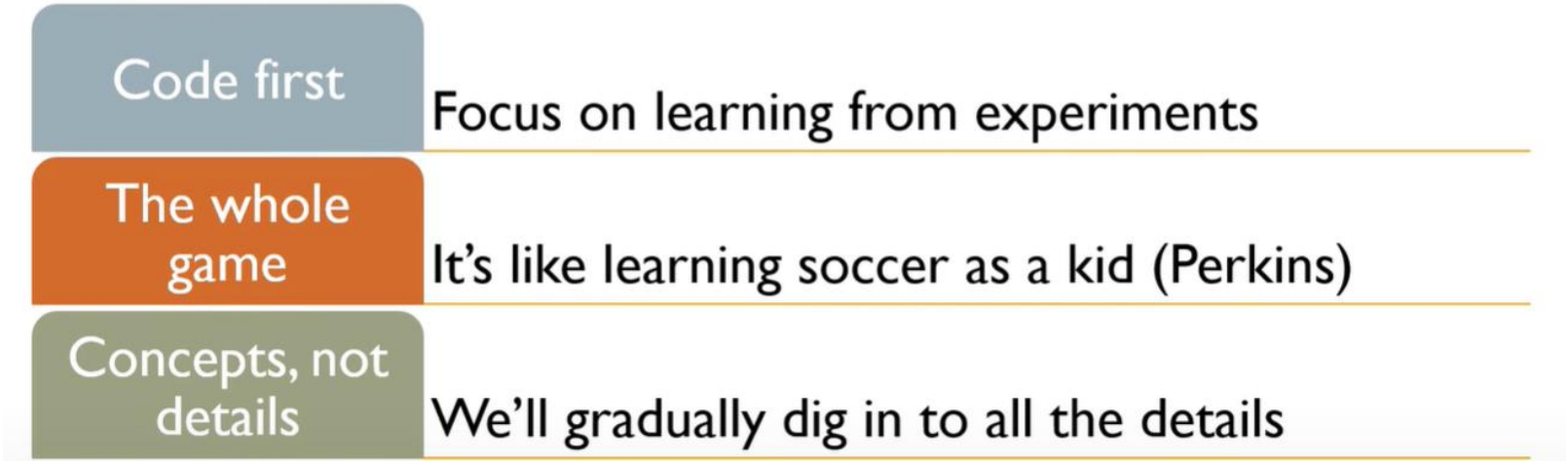
State of the art Anomaly Detection with Anomalib

ANOMALiB

AI is magic, I will never understand it.

Black box	• Interpretable ML • Visualize gradients and activations
Needs too much data	• Transfer learning • Share pre-trained nets
Needs ML PhD	• No longer true • fastai & keras libs, MOOCs, etc
Only for vision	• No longer true • SoTA for speech, structured data, time series…
Needs lots of GPUs	• Was never true • …except for some research projects
"Not really AI"	• Who cares? • Do you really want to build a brain?

Code first approach (TopDown)



You will learn how it works before you learn why it works!

Where is deep learning the best known approach?

NLP	answering questions; speech recognition; summarizing documents; classifying documents; finding names, dates, etc. in documents; searching for articles mentioning a concept
Computer vision	satellite and drone imagery interpretation (e.g. for disaster resilience); face recognition; image captioning; reading traffic signs; locating pedestrians and vehicles in autonomous vehicles
Medicine	finding anomalies in radiology images, including CT, MRI, and x-ray; counting features in pathology slides; measuring features in ultrasounds; diagnosing diabetic retinopathy
Biology	folding proteins; classifying proteins; many genomics tasks, such as tumor-normal sequencing and classifying clinically actionable genetic mutations; cell classification; analyzing protein/protein interactions
Image generation	colorizing images; increasing image resolution; removing noise from images; converting images to art in the style of famous artists
Recommendation systems	web search; product recommendations; home page layout
Playing games	better than humans and better than any other computer algorithm at Chess, Go, most Atari videogames, many real-time strategy games
Robotics	handling objects that are challenging to locate (e.g. transparent, shiny, lack of texture) or hard to pick up
Other applications	financial and logistical forecasting; text to speech; much much more...

How long you train.

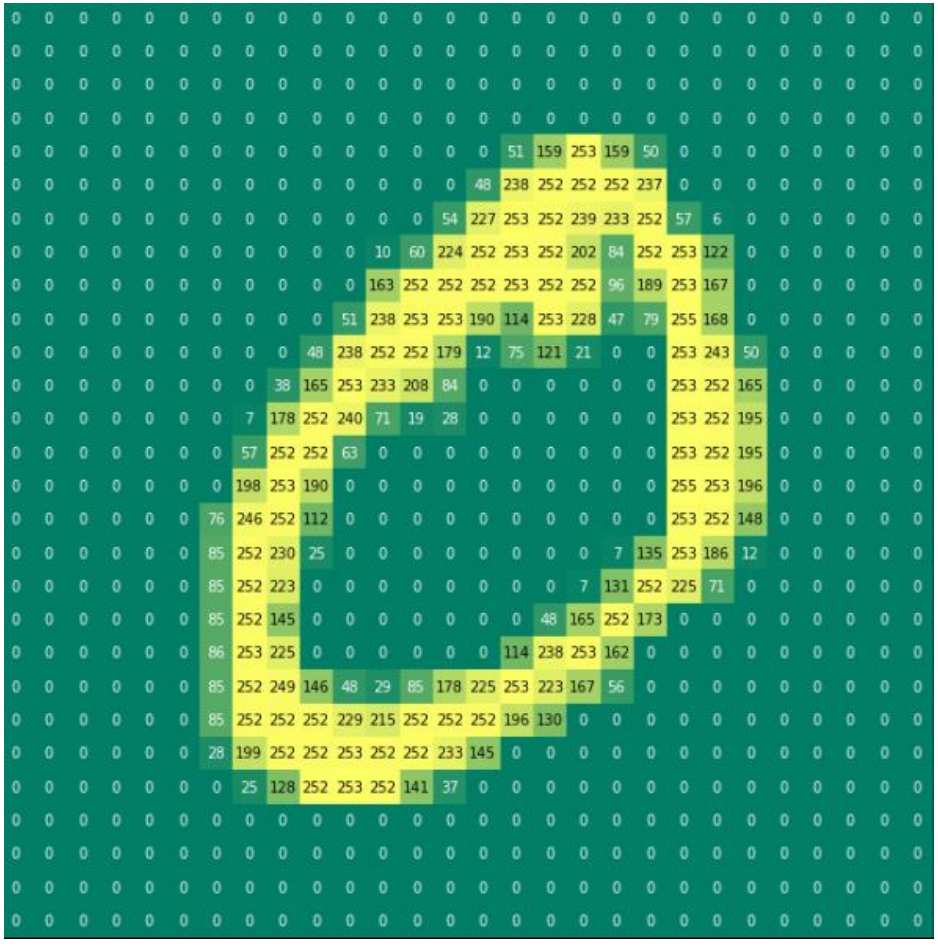
ImageNet dataset ~1.6 Mio pictures

Konfiguration	Float 32 Training	Float 16 Training
CPU(32 cores)	27 Stunden	27 Stunden
Single RTX 2080TI	69 Minuten	29 Minuten
Single RTX 3080	53 Minuten	22 Minuten
Single RTX 3090	41 Minuten	18 Minuten
Single Tesla A100	23 Minuten	8,5 Minuten
4 x RTX 2080TI	19 Minuten	8 Minuten
4 x Tesla V100	15 Minuten	4,5 Minuten
4 x RTX 3090	11,5 Minuten	5 Minuten
4 x Tesla A100	6,5 Minuten	3 Minuten

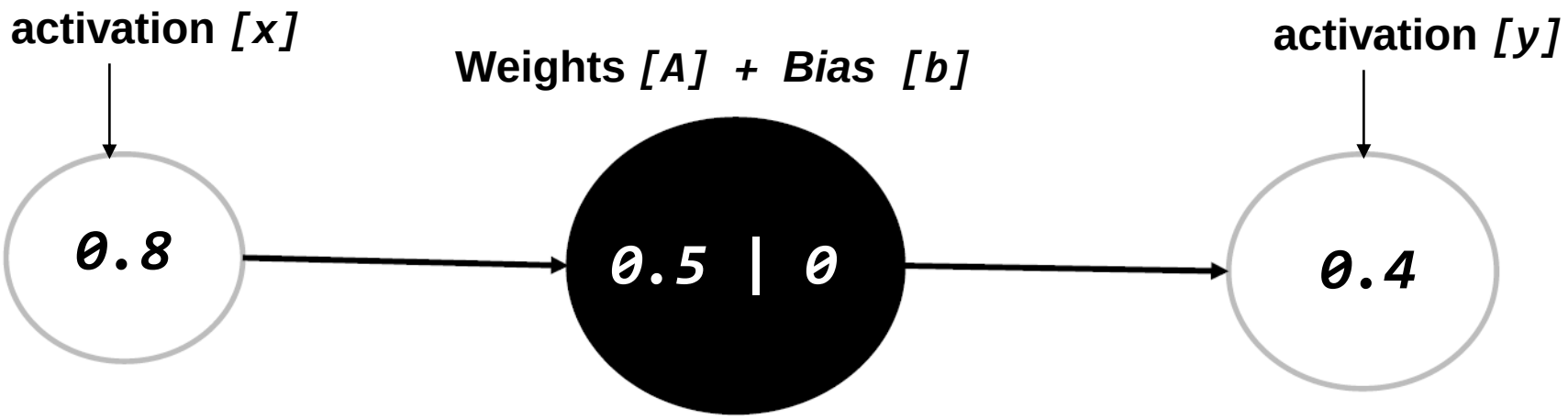
Lab session

What is machine learning / deep learning

How a computer sees numbers

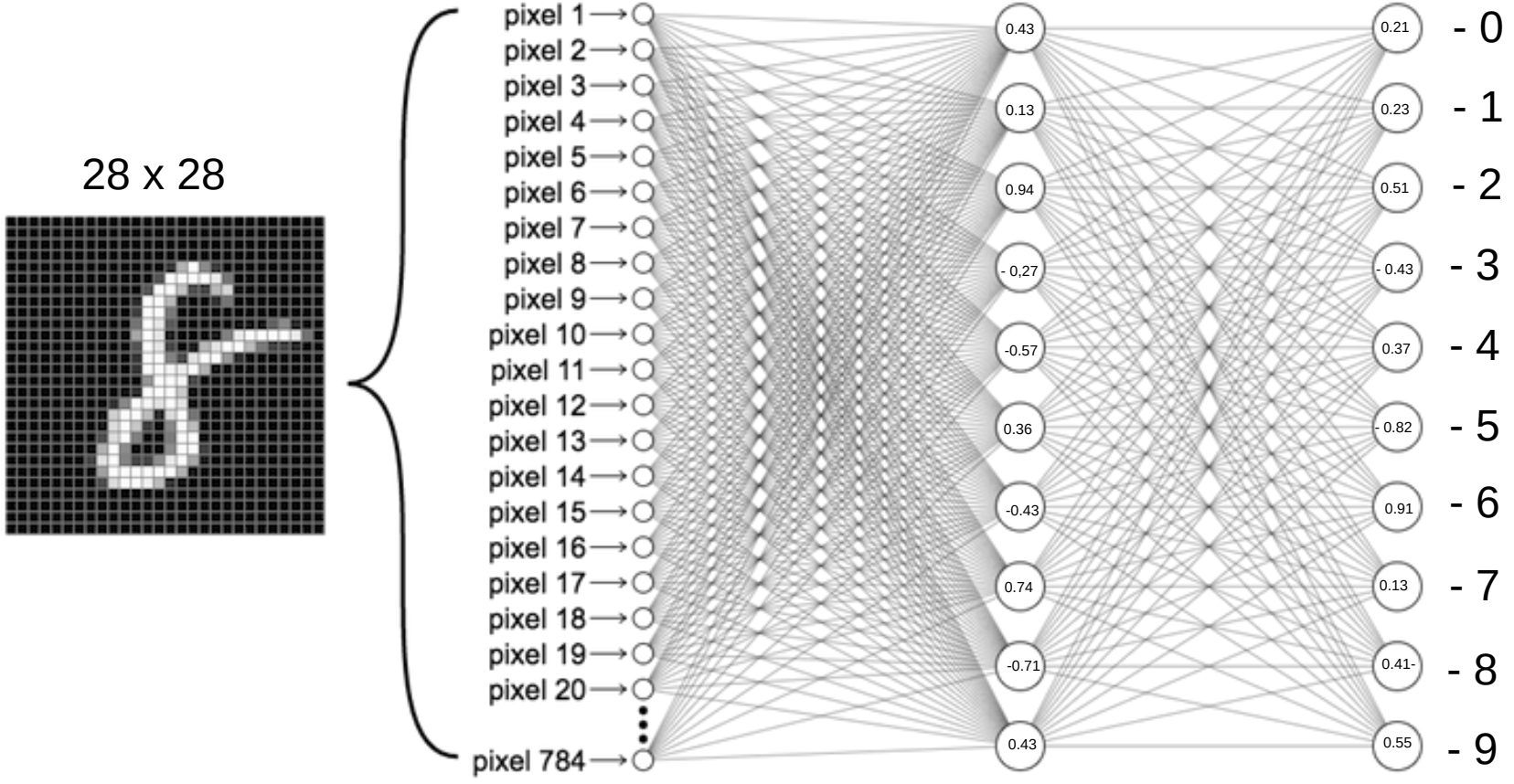


Neuron



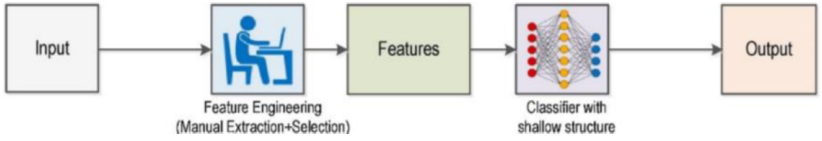
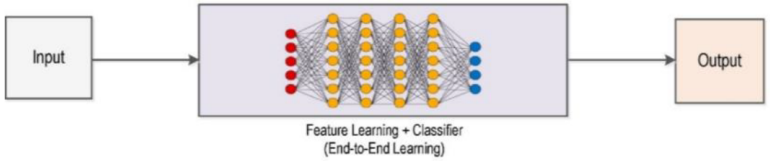
$$y = x * A + b$$

Forward Propagation



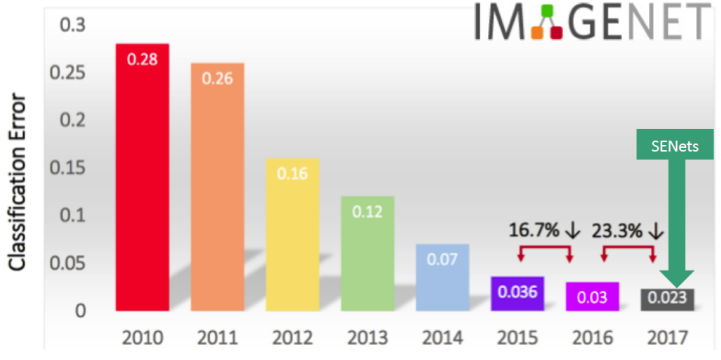
Backward Propagation

Deep learning vs. classic machine learning

Traditional CV / ML	Deep Learning
	
• Works better on small data • Financially and computationally cheap • Easier to interpret	• Best-in-class performance • Scales effectively with data • No need for feature engineering • Adaptable and transferable

ImageNet

- 15 million of labeled images covering almost 22,000 categories.
- **I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge (**ILSVRC**) uses a smaller portion of ImageNet (1000 categories, 1.3 million images)



learning methods

Supervised Learning

- Regression
  - *Predictions*
  - *Optimizations*
  - *Modeling*
  - *Forecasts*
- Classification
  - *Diagnosis*
  - *Sorting*
  - *Recommender System*
- *Algorithms*
  - *Linear Regression*
  - *Logistic Regression*
  - *Random Forests*
  - *Multilayer Percept.*
  - *CNN*
  - *RNN*
  - *…..*

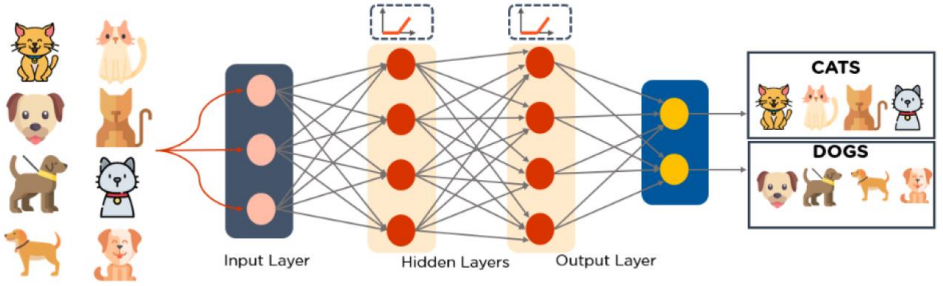
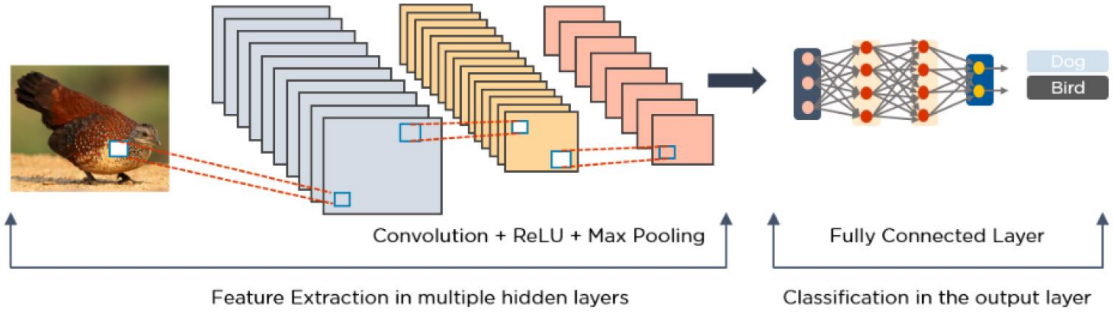
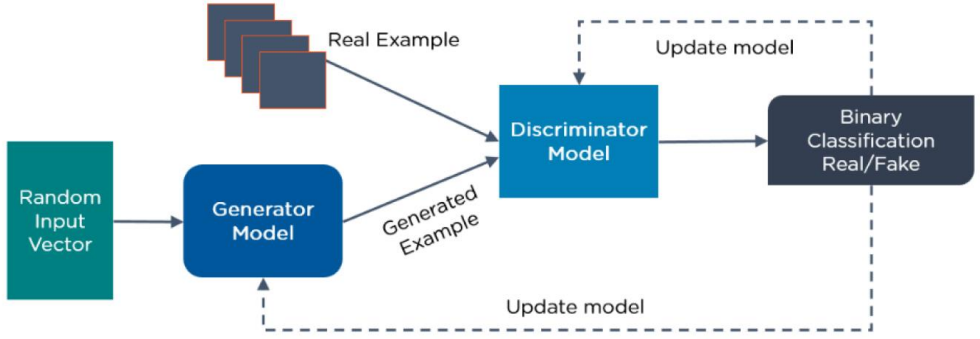
Unsupervised Learning

- Clustering
  - *Grouping*
  - *Pattern recognition*
- Dimension Reduction
- Anomaly Detection
- Algorithms
  - K-Means
  - PCA
  - Gaussian Mixture
  - Singular value decomposition
  - Autoencoder

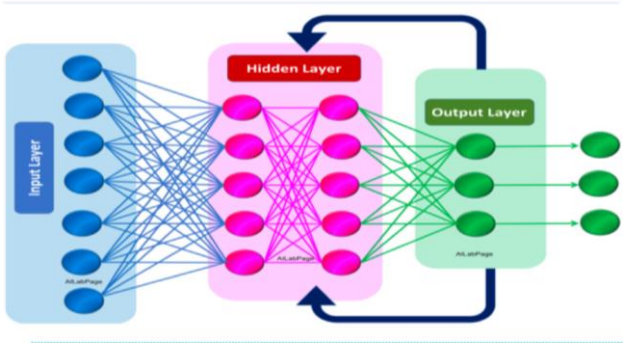
Reinforcement Learning

- Game Intelligence (Alpha Go)
- Swarm Intelligence
- Machine optimization
- Algorithms
  - Monte Carlo
  - Deep Q-learning
  - State–action–reward–state–action (SARSA)

Famous Neural-Net architectures 1

1. Multilayer Perceptrons (**MLPs**)	
2. Convolutional Neural Networks (**CNNs**)	
3. Generative Adversarial Networks (**GANs**)	

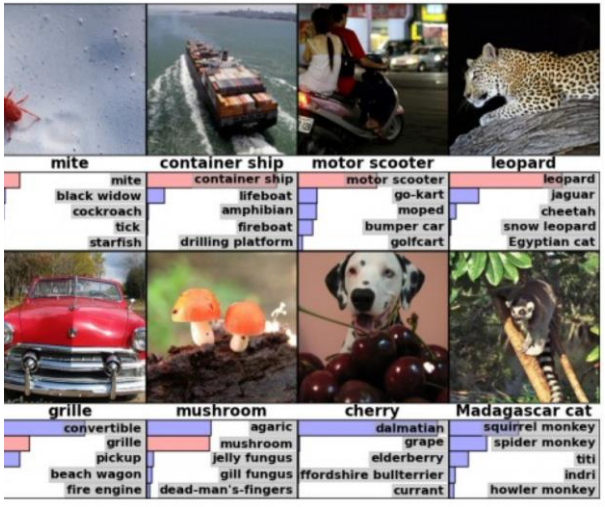
Famous Neural-Net architectures 2

4. Recurent Neural Networks (**RNN**)	
5. Transformer Networks (**TN**)	

Convolutional Neural Networks (CNN) 1/4

2012 to Present: CNN's are everywhere

Image Classification



Object Detection

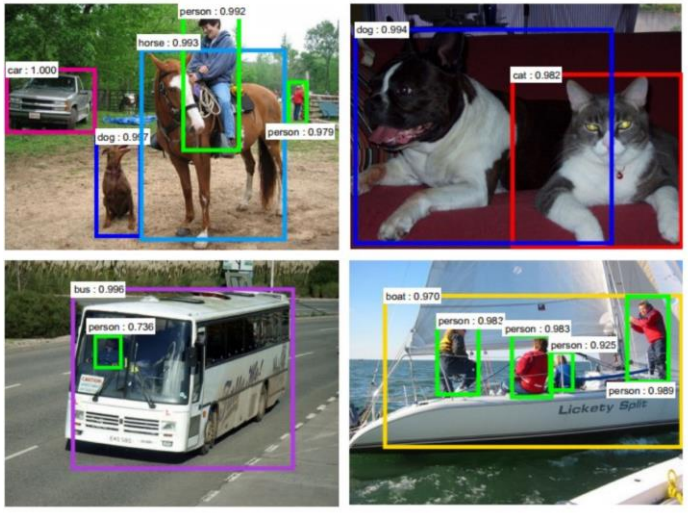


Image Retrieval

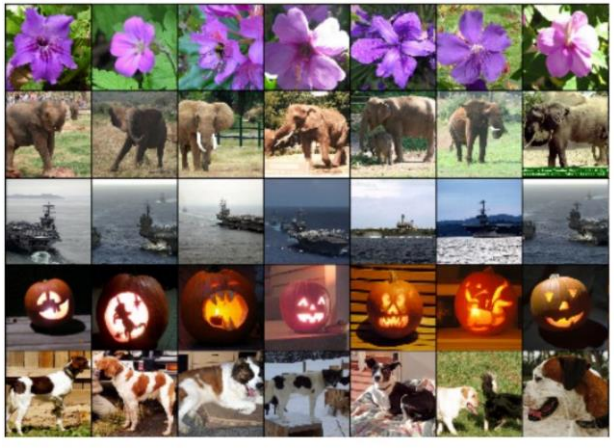
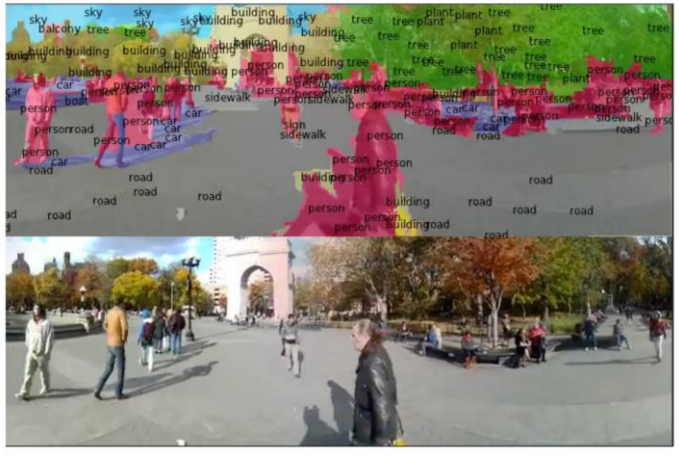


Image Segmentation



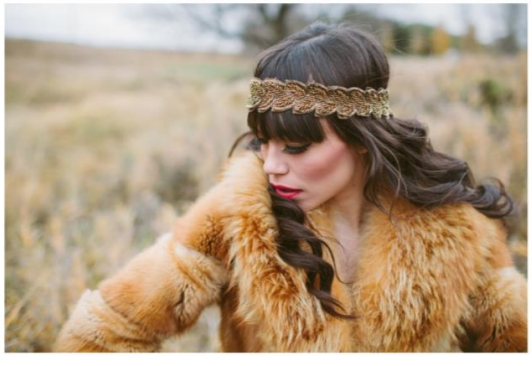
2012 to Present: CNN's are everywhere

Pose Recognition

Image Captioning

A white teddy bear sitting in the grass

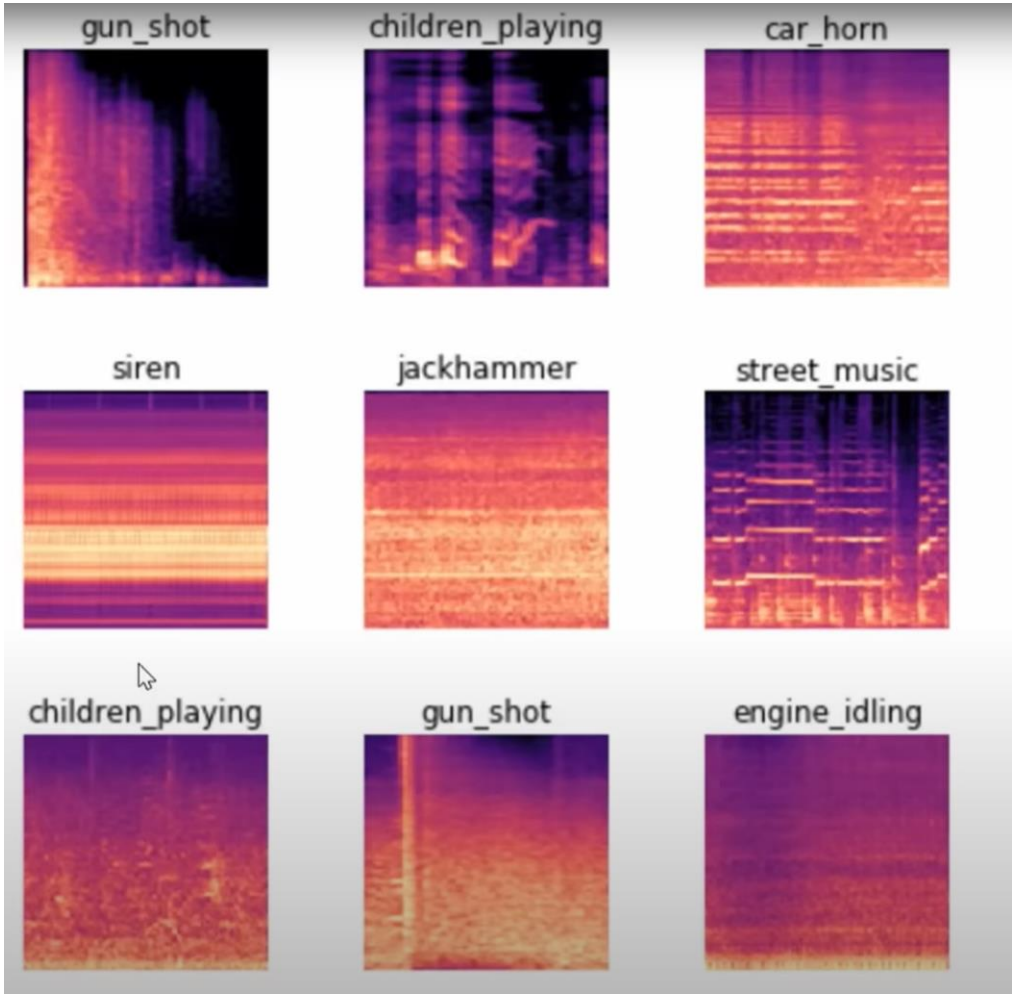
A man in a baseball uniform throwing a ball



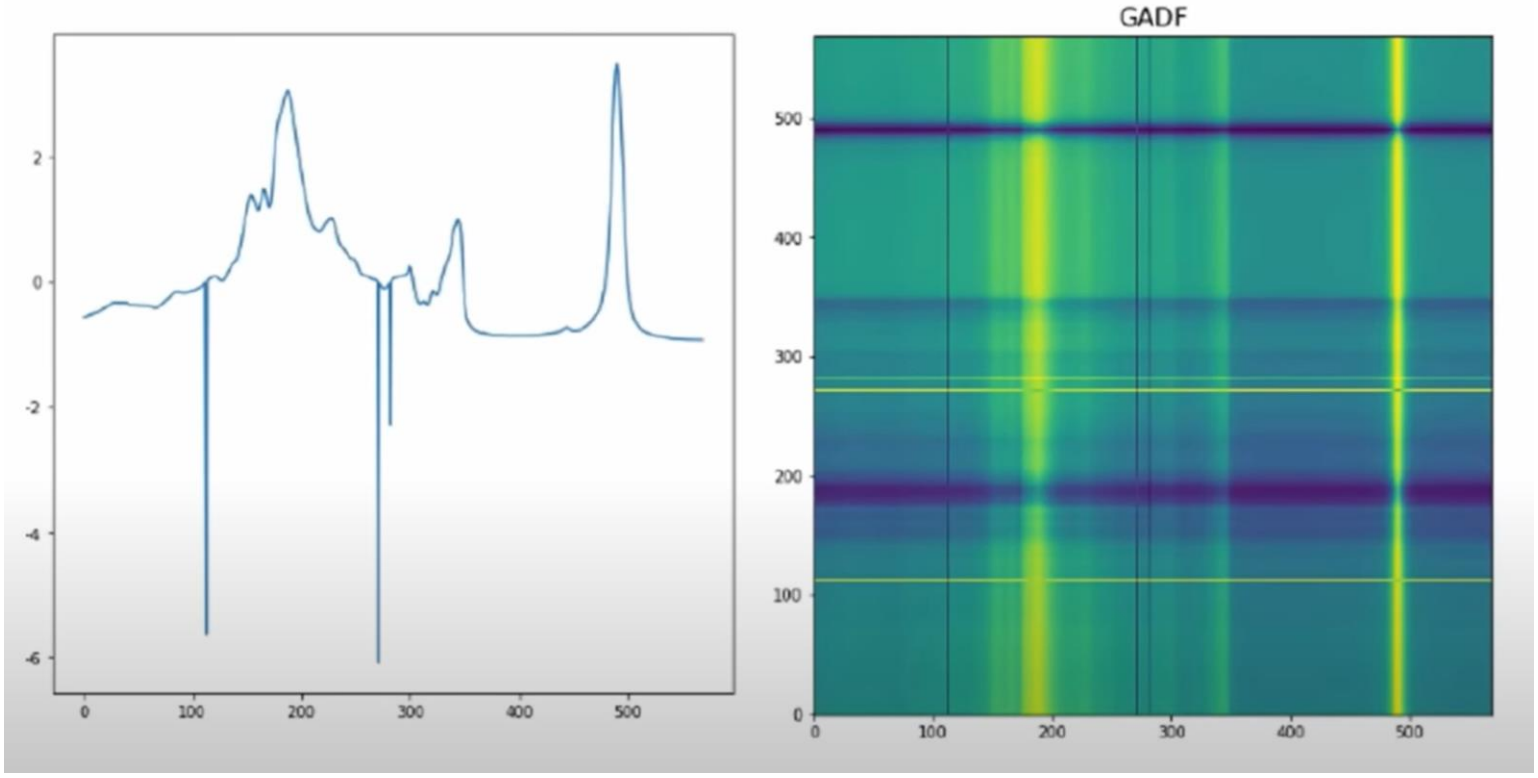
A woman is holding a cat in her hand

2012 to Present: CNN's are everywhere

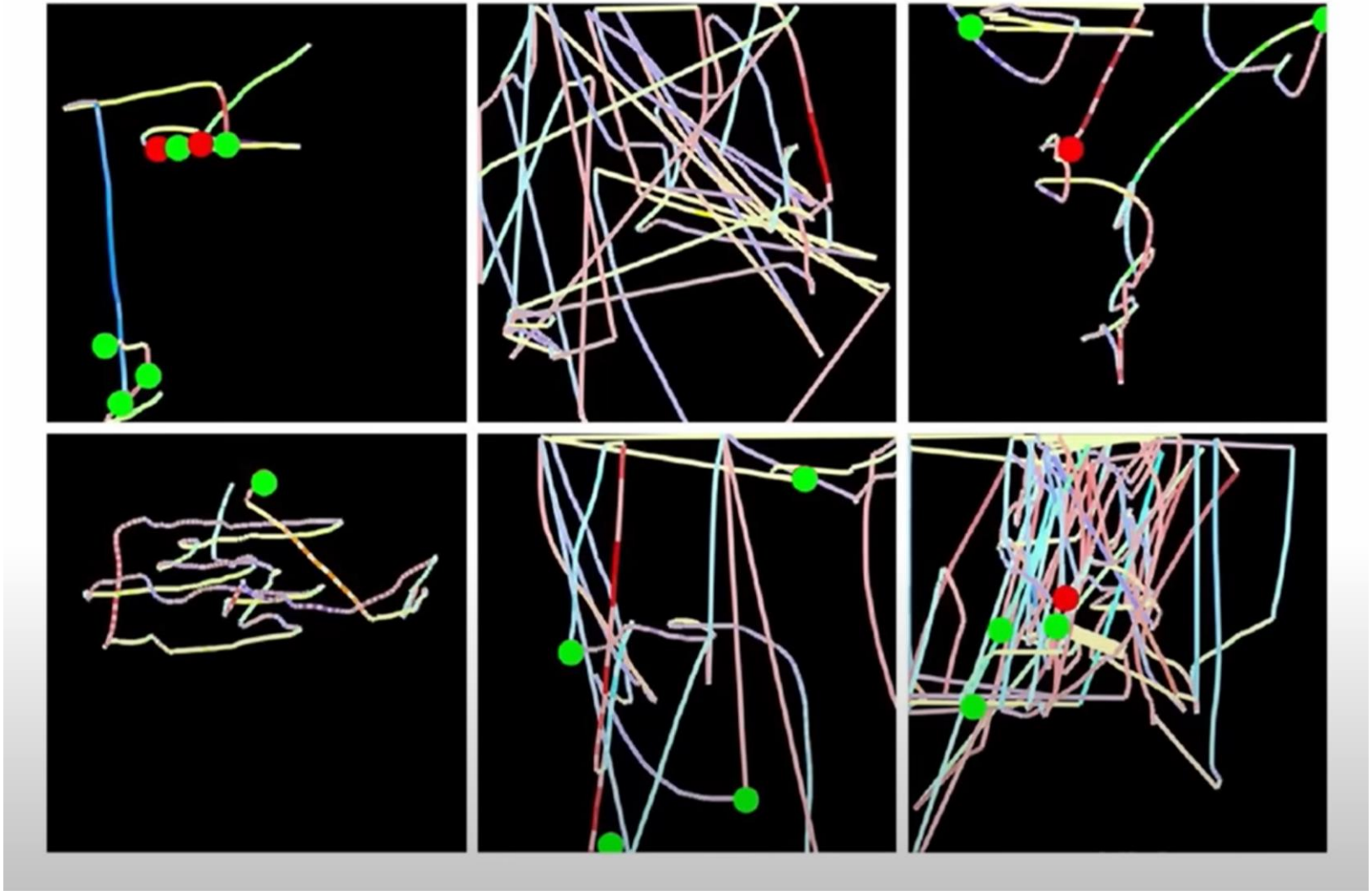
Exotic 1


gun_shot
children_playing
car_horn
siren
jackhammer
street_music
children_playing
gun_shot
engine_idling

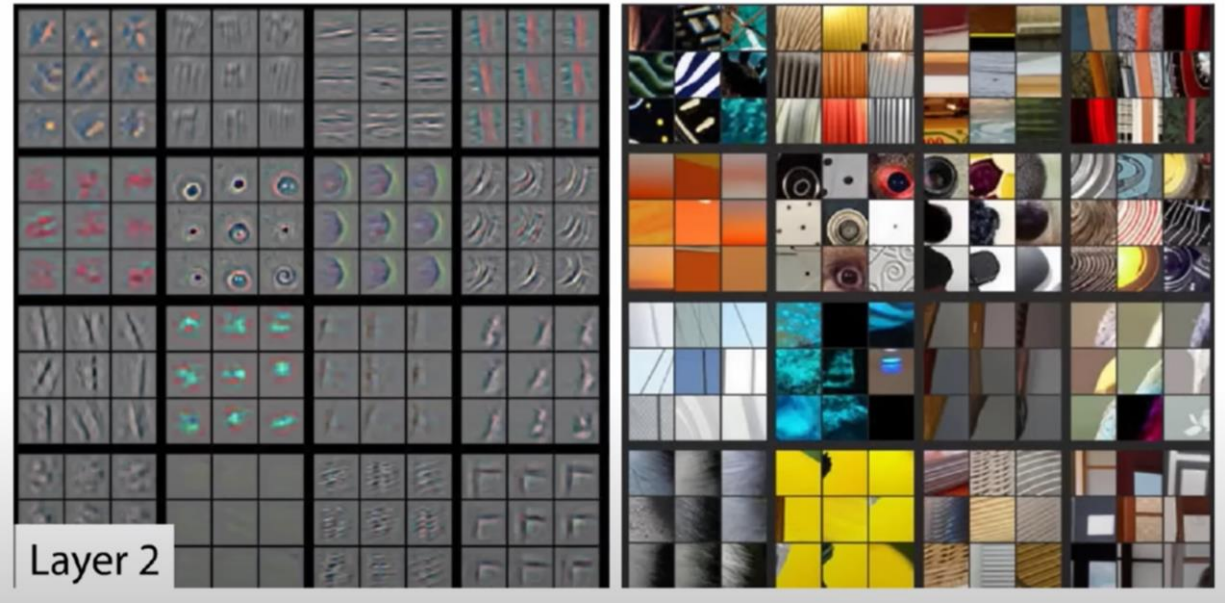
Exotic 2


2
0
-2
-4
-6
0
100
200
300
400
500
GADF
500
400
300
200
100
0
0
100
200
300
400
500

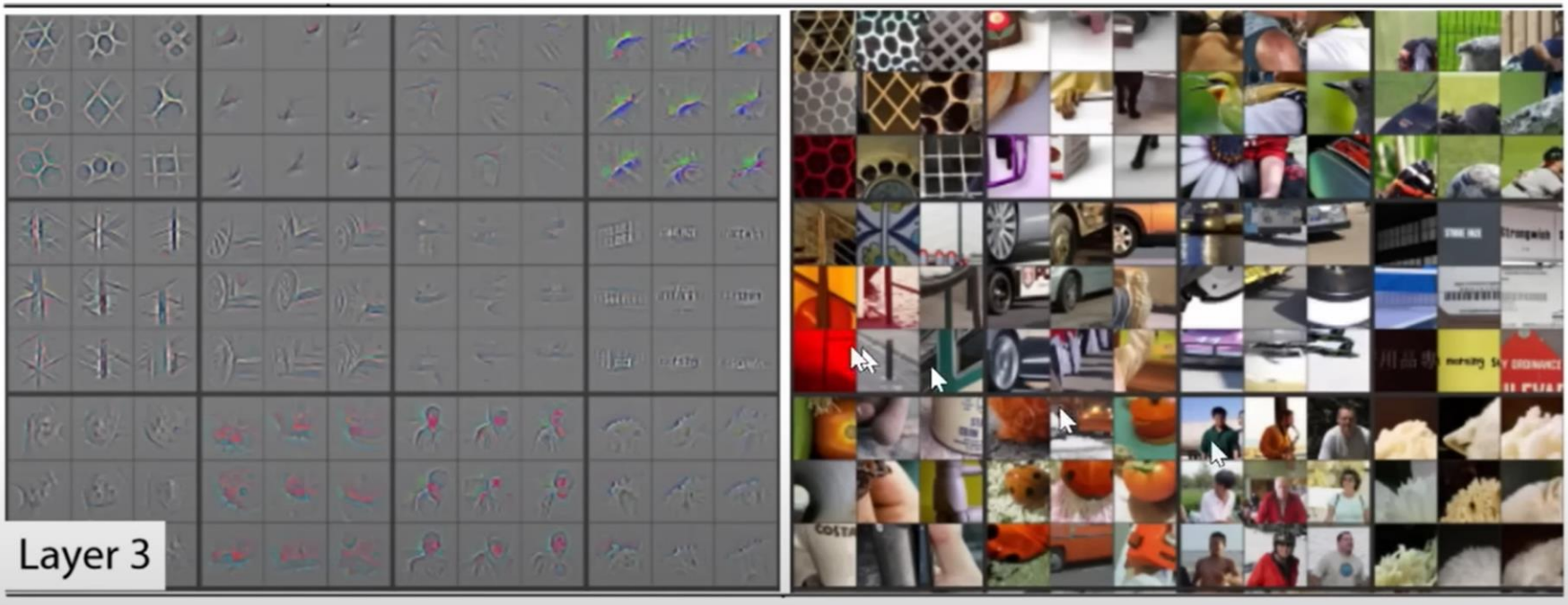
Exotic 3



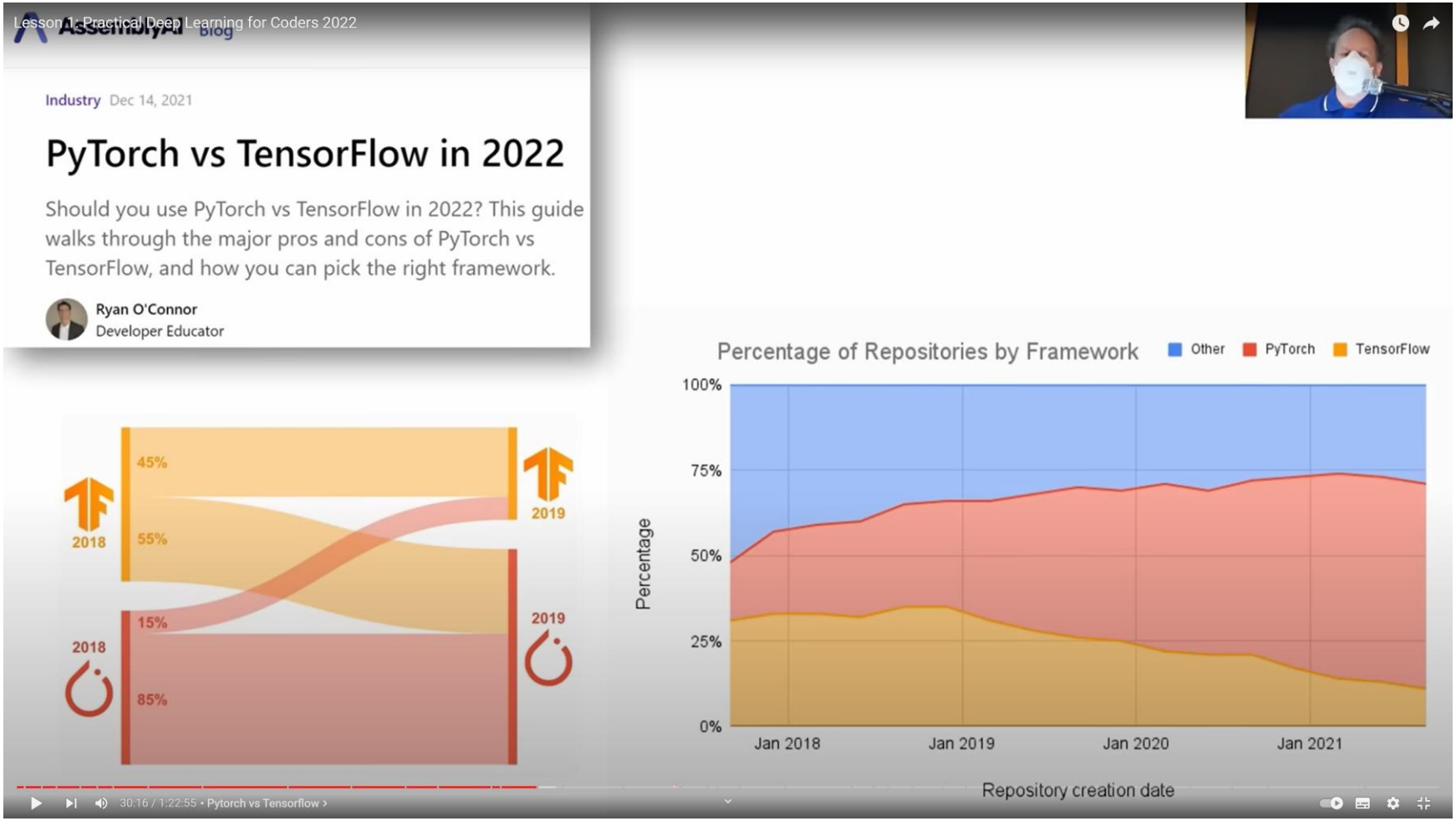
How CNN works



How CNN works



Why we use Pytorch and FastAI



Why we use Pytorch and FastAI

Here's AdamW's step in PyTorch.

```
loss = None
if closure is not None:
    loss = closure()

for group in self.param_groups:
    for p in group['params']:
        if p.grad is None:
            continue

        # Perform stepweight decay
        p.data.mul_(1 - group['lr'] * group['weight_decay'])

        # Perform optimization step
        grad = p.grad.data
        if grad.is_sparse:
            raise RuntimeError('Adam does not support sparse gradients, please consider SparseAdam instead')
        amsgrad = group['amsgrad']

        state = self.state[p]

        # State initialization
        if len(state) == 0:
            state['step'] = 0
            # Exponential moving average of gradient values
            state['exp_avg'] = torch.zeros_like(p.data)
            # Exponential moving average of squared gradient values
            state['exp_avg_sq'] = torch.zeros_like(p.data)
            if amsgrad:
                # Maintains max of all exp. moving avg. of sq. grad. values
                state['max_exp_avg_sq'] = torch.zeros_like(p.data)

        exp_avg, exp_avg_sq = state['exp_avg'], state['exp_avg_sq']
        if amsgrad:
            max_exp_avg_sq = state['max_exp_avg_sq']
        beta1, beta2 = group['betas']

        state['step'] += 1

        # Decay the first and second moment running average coefficient
        exp_avg.mul_(beta1).add_(1 - beta1, grad)
        exp_avg_sq.mul_(beta2).addcmul_(1 - beta2, grad, grad)
        if amsgrad:
            # Maintains the maximum of all 2nd moment running avg. till now
            torch.max(max_exp_avg_sq, exp_avg_sq, out=max_exp_avg_sq)
            # Use the max. for normalizing running avg. of gradient
            denom = max_exp_avg_sq.sqrt().add_(group['eps'])
        else:
            denom = exp_avg_sq.sqrt().add_(group['eps'])

        bias_correction1 = 1 - beta1 ** state['step']
        bias_correction2 = 1 - beta2 ** state['step']
        step_size = group['lr'] * math.sqrt(bias_correction2) / bias_correction1

        p.data.addcdiv_(-step_size, exp_avg, denom)

return loss
```

Why we use Pytorch and FastAI

Lesson 1: Practical Deep Learning for Coders 2022

Zum Beenden des Vollbildmodus Esc drücken

```python
def weight_decay(p, lr, wd, do_wd=True, **kwargs):
    "Weight decay as decaying `p` with `lr*wd`"
    if do_wd: p.data.mul_(1 - lr*wd)
    return p
weight_decay.defaults = dict(wd=0.)
```

```python
"A `Optimizer` for Adam with `lr`, `mom`, `sqr_mom`, `eps` and `params`"
steppers = [] if wd==0. else [weight_decay] if decouple_wd else [l2_reg]
steppers.append(adam_step)
stats = [partial(average_grad, dampening=True), average_sqr_grad, step_stat]
return Optimizer(params, steppers, stats=stats, lr=lr, mom=mom, sqr_mom=sqr_mom, eps=eps, wd=wd)
```

In fastai it's very different

Lab session