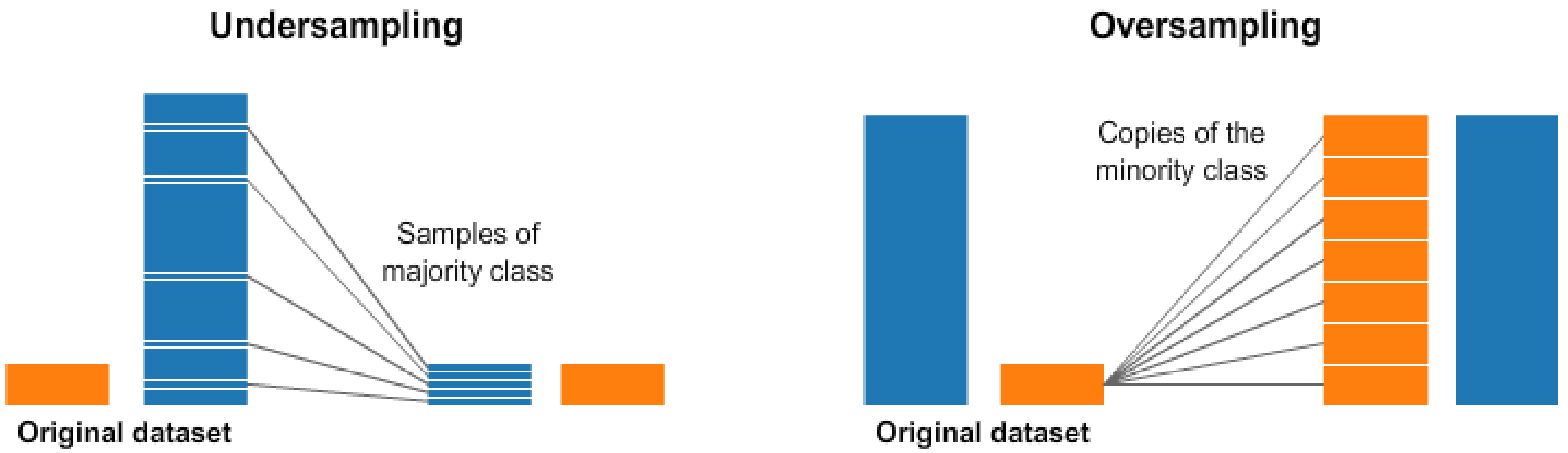
Model Development

How to deal with class imbalance

- Add more minority samples or remove majority samples (resampling)
- Adjust losses (weight balancing)
- Choose algorithms robust to class imbalance

Class imbalance solution: Resampling

Undersampling	**Oversampling**
Remove samples from the majority class	Add more examples to the minority class
Can cause loss of information	Can cause overfitting



https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets#t1

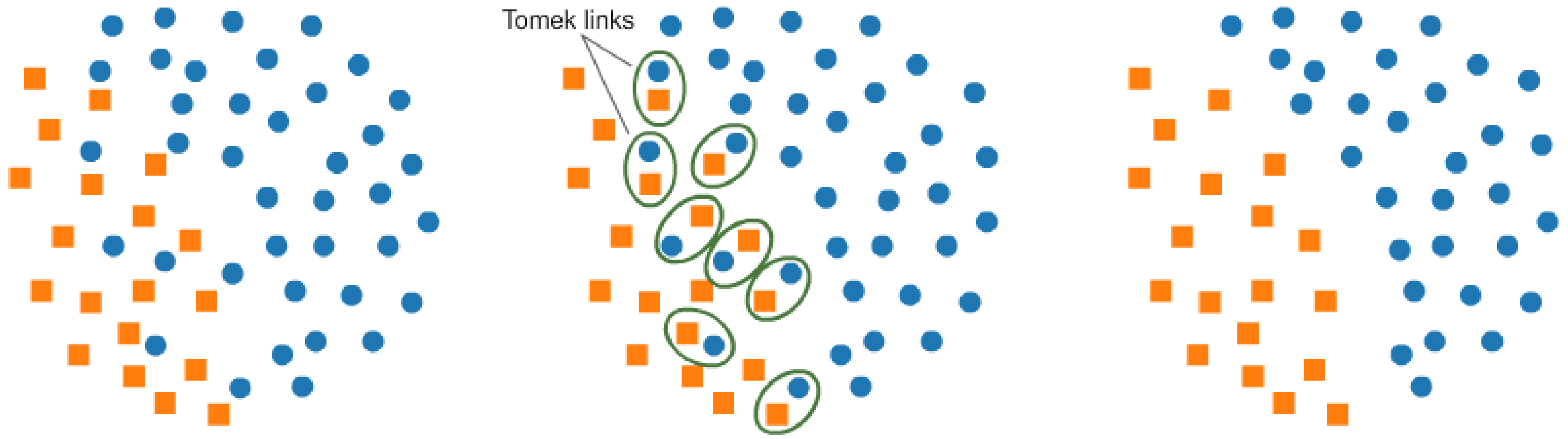
Class imbalance solution: Resampling

- Undersampling can be done by random sampling or more sophisticated methods could be used:
  - For example: cluster the records of the majority class, and do the under-sampling by removing records from each cluster, to preserve information.
- In over-sampling, instead of creating exact copies of the minority class records, we can introduce small variations into those copies, creating more diverse synthetic samples.
- Python library imbalanced-learn is compatible with scikit-learn and is part of scikit-learn-contrib projects.
  - Some examples: https://imbalanced-learn.org/stable/auto_examples/index.html#general-examples

https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets#t1

Undersampling: Tomek Links

- Find pairs of close samples of opposite classes
- Remove the sample of majority class in each pair
  - Pros: Make decision boundary more clear
  - Cons: Make model less robust



https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Oversampling: SMOTE

- Synthesize samples of minority class as convex (~linear) combinations of existing points and their nearest neighbors of same class.

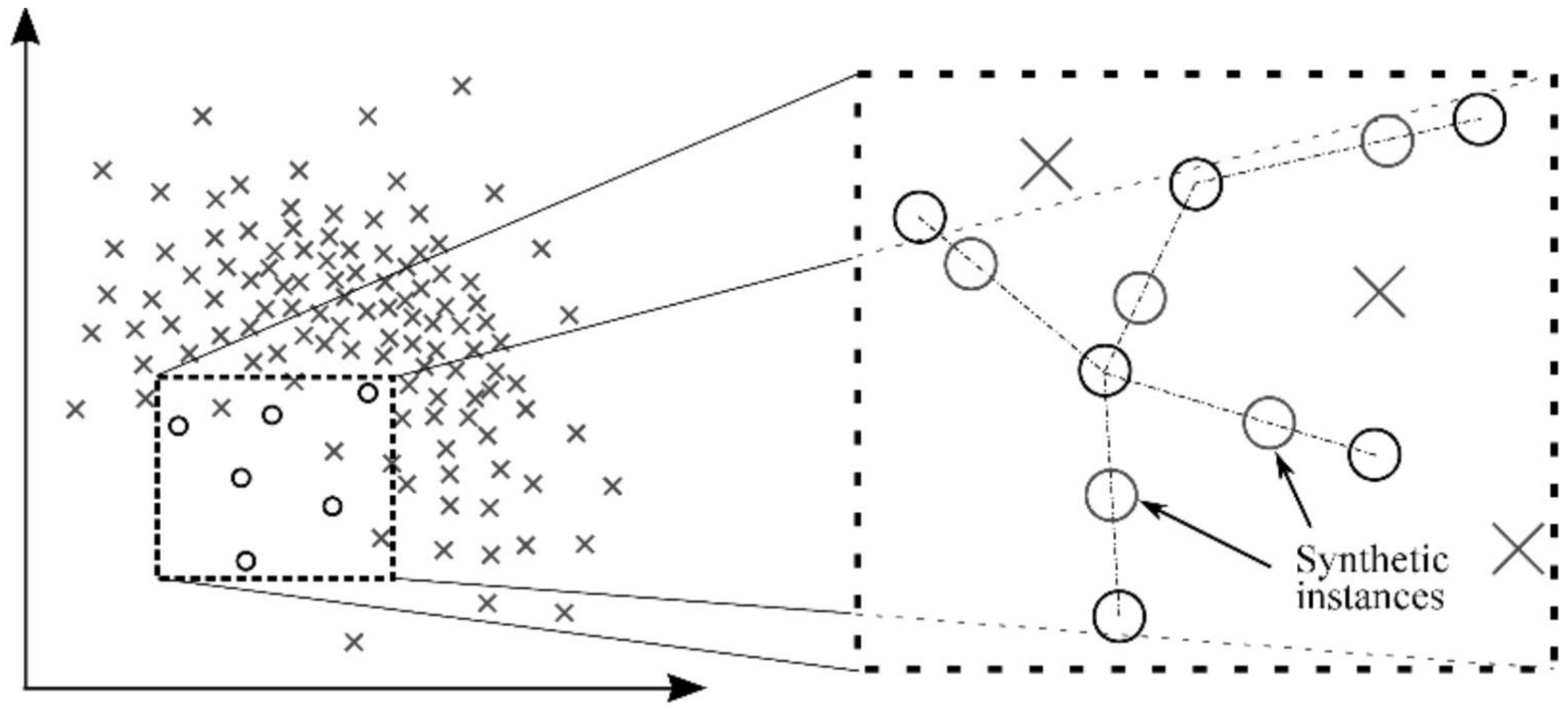


Image from [Analytics Vidhya]()

Class imbalance solution: weight balancing

- Give more weight to rare classes

Non-weighted loss

$$L(X;\ \theta) = \sum_i L(x_i;\theta)$$

Weighted loss

$$L(X;\ \theta) = \sum_i W_{y_i} L(x_i;\theta)$$

$$W_c = \frac{N}{\textit{number of samples of class } C}$$

```
model.fit(features, labels, epochs=10, batch_size=32, class_weight={“fraud”: 0.9, “normal”: 0.1})
```

Class imbalance solution: weight balancing

- Give more weight to difficult samples:
  - downweighs well-classified samples

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$

$$\mathrm{CE}(p_t) = -\log(p_t)$$

$$\mathrm{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

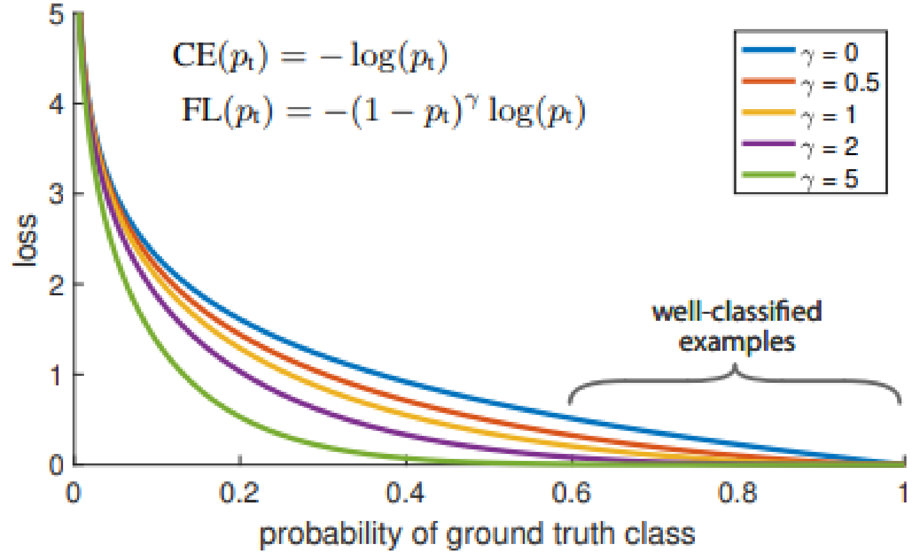


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

Focal Loss for Dense Object Detection (Lin et al., 2017)

Class imbalance solution: algorithms

Bagging (Bootstrap Aggregating)

- Sample with replacement to create different datasets
- Train a classifier with each dataset
- Aggregate predictions from classifiers

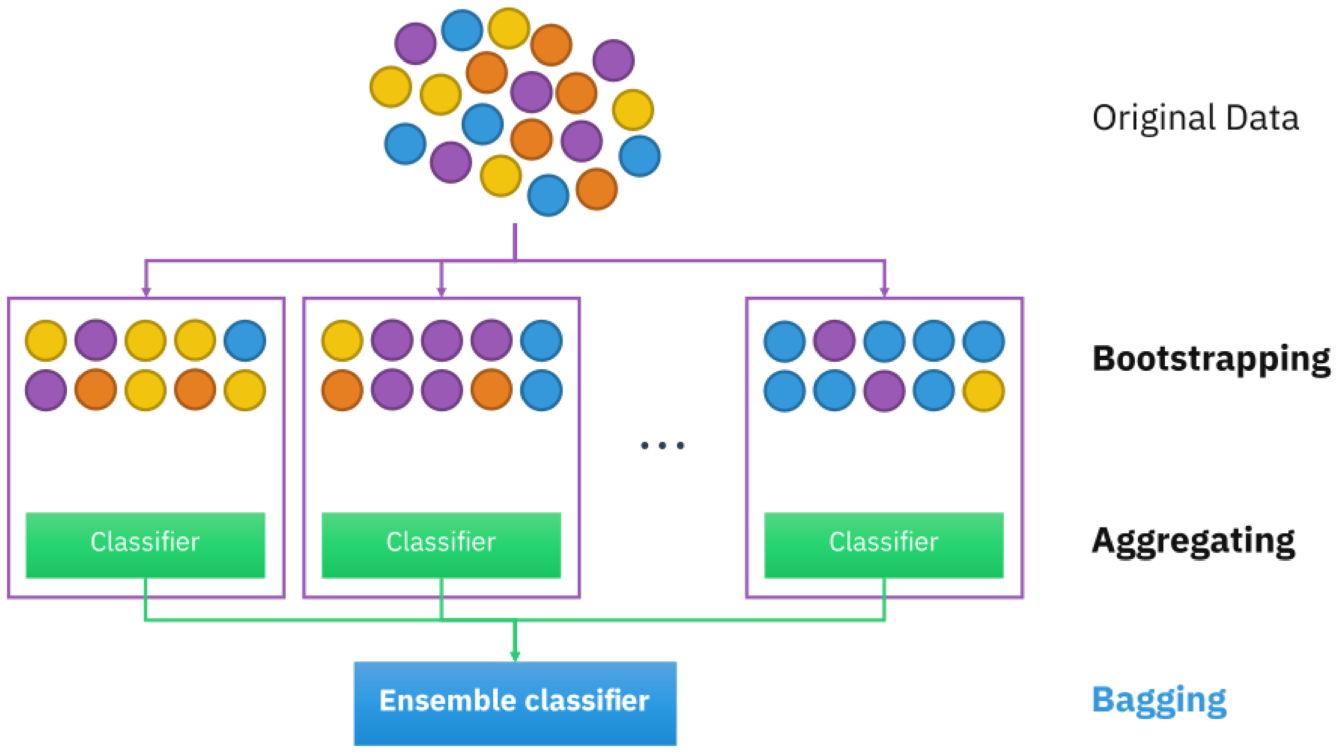


Illustration by Sirakorn

Class imbalance solution: algorithms

Boosting

1. Train a weak classifier
2. Give samples misclassified by the weak classifier more weight
3. Repeat (1) on this reweighted data

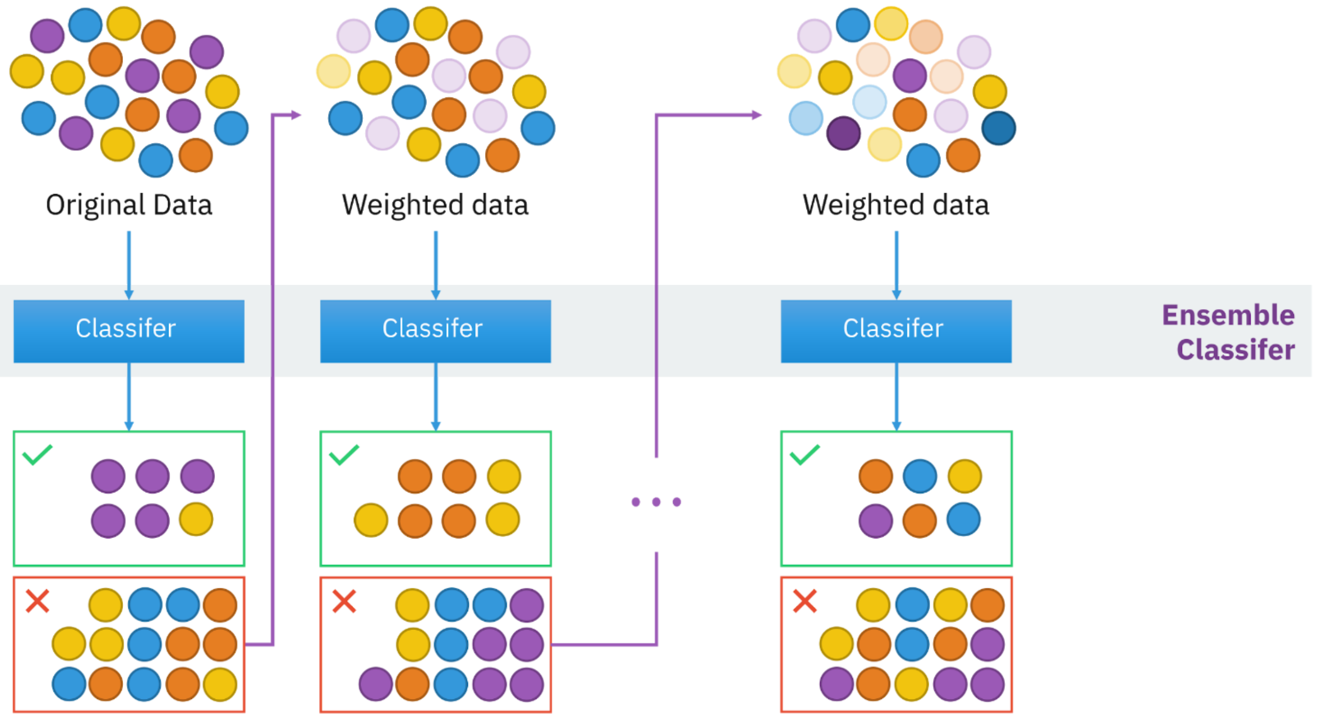


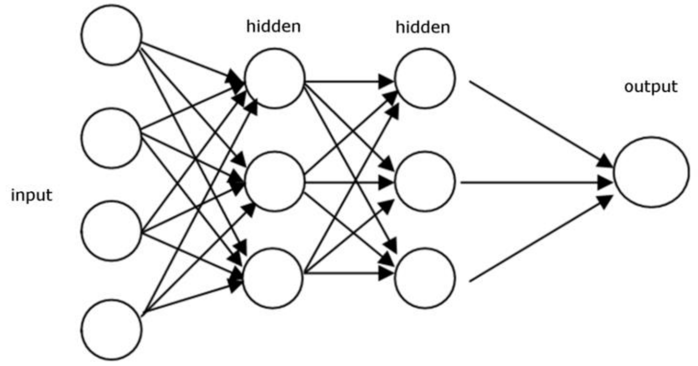
Illustration by [Sirakorn]

Class imbalance solution: algorithms

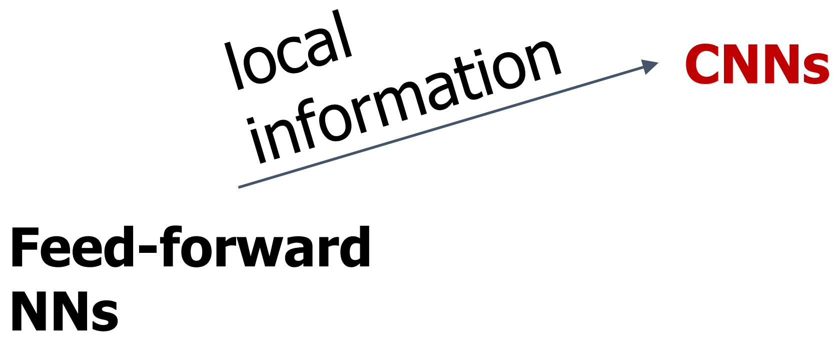
- An example of a boosting algorithm is Gradient Boosting Machine which produces a prediction model typically from weak decision trees.
- It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.
- XGBoost, a variant of GBM: an algorithm used by many winning teams of machine learning competitions for a wide range of tasks.
- Recently, LightGBM, a distributed gradient boosting framework that allows parallel learning which generally allows faster training on large datasets, is being used by many teams.

Neural architecture evolution

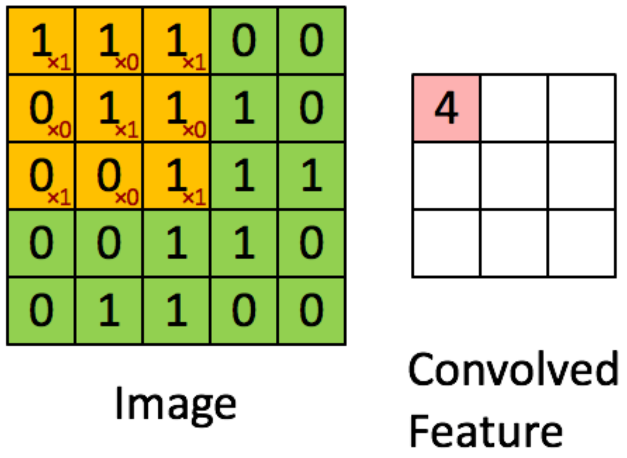
Feed-forward NNs

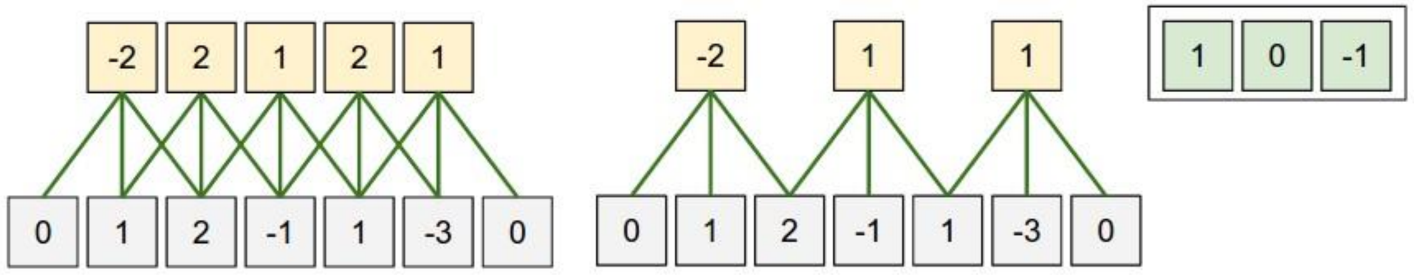


Neural architecture evolution



Instead of connecting each neuron to all input, connect each neuron to a subset of inputs



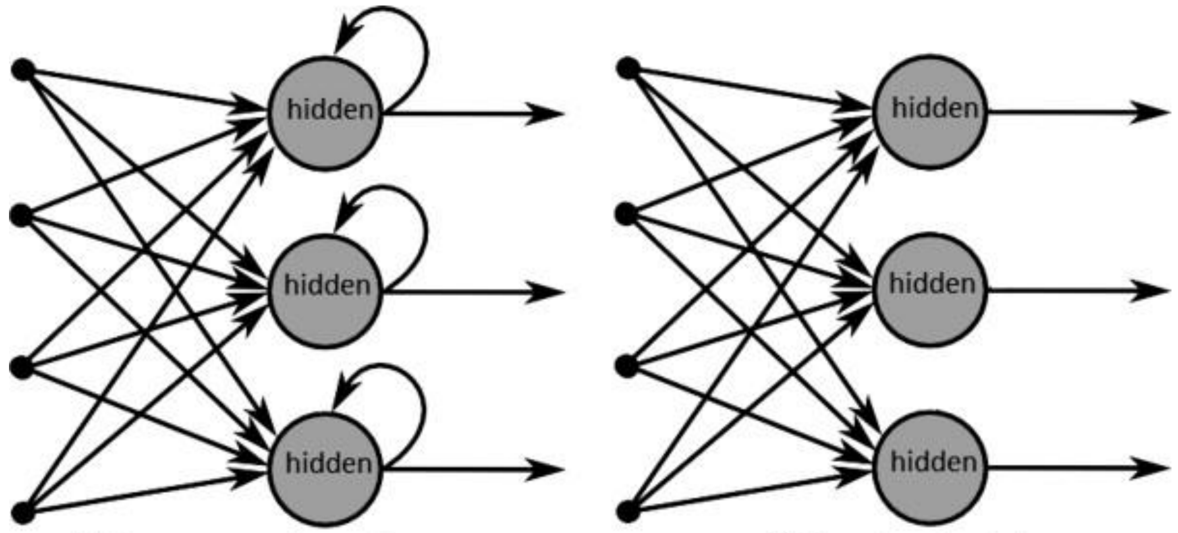


Left image from http://deeplearning.stanford.edu/, right image from S231N Lecture 7.

Neural architecture evolution

$$\begin{tikzcd}
 & \textbf{CNNs} \\
\textbf{Feed-forward NNs} \arrow[ur, "\text{local information}"] \arrow[dr, "\text{weights sharing across steps}"'] & \\
 & \textbf{RNNs}
\end{tikzcd}$$

Instead of using different weights for different timesteps, use same weights!

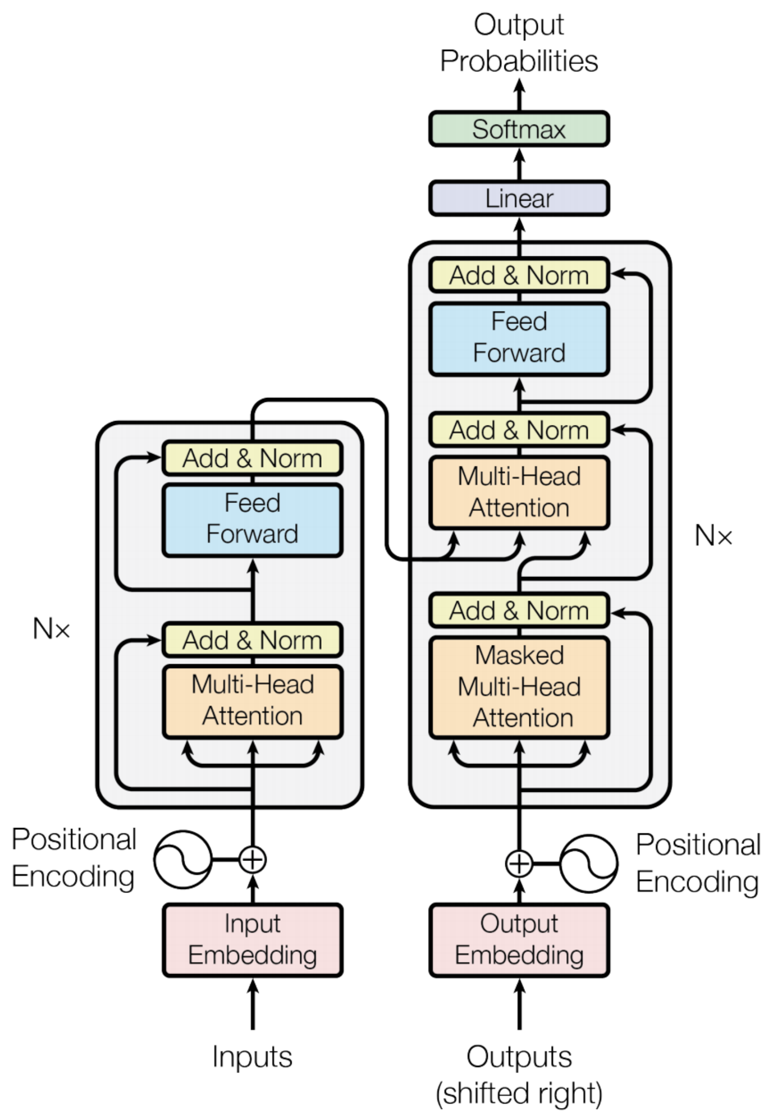


(a) Recurrent neural network (b) Forward neural network

"A survey on the application of recurrent neural networks to statistical language modeling." (De Mulder et al., 2015)

Neural architecture evolution

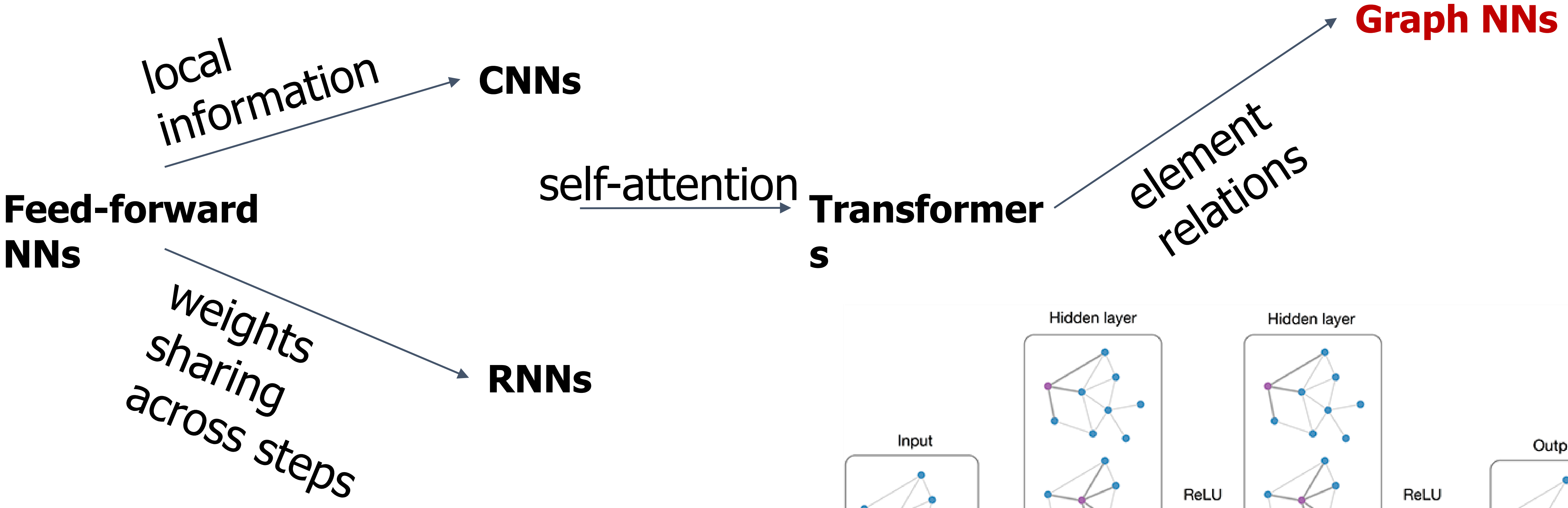
$$\begin{tikzcd}
 & \textbf{CNNs} \\
\textbf{Feed-forward NNs} \arrow[ur, "\text{local information}"] \arrow[dr, "\text{weights sharing across steps}"'] & \text{self-attention} \xrightarrow{} \textbf{Transformers} \\
 & \textbf{RNNs}
\end{tikzcd}$$

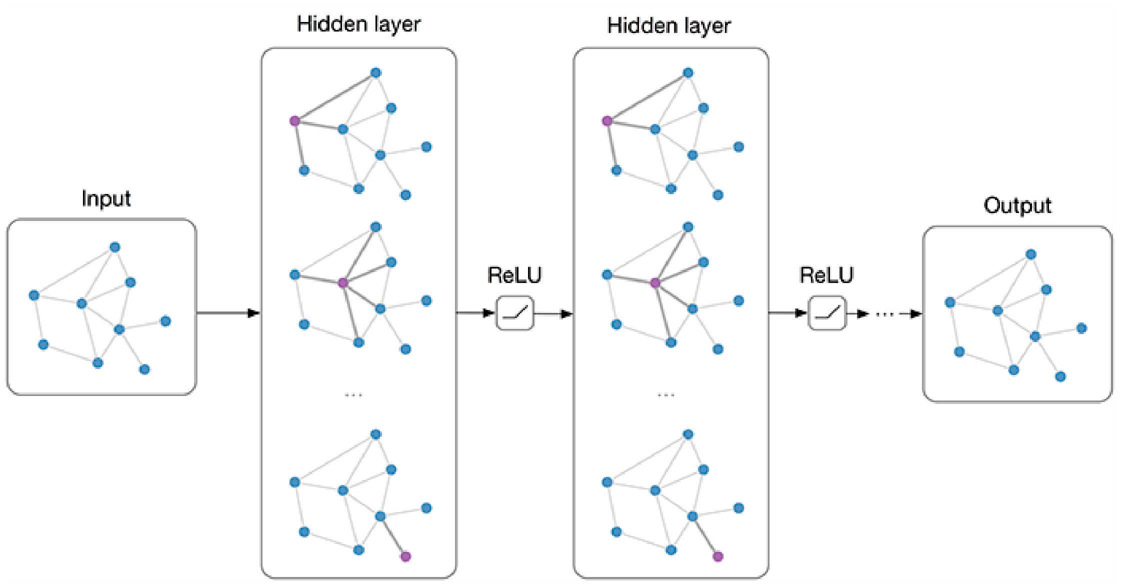


Allow classifier to peak at all previous hidden states

Attention Is All You Need (Vaswani et al., 2017)

Neural architecture evolution



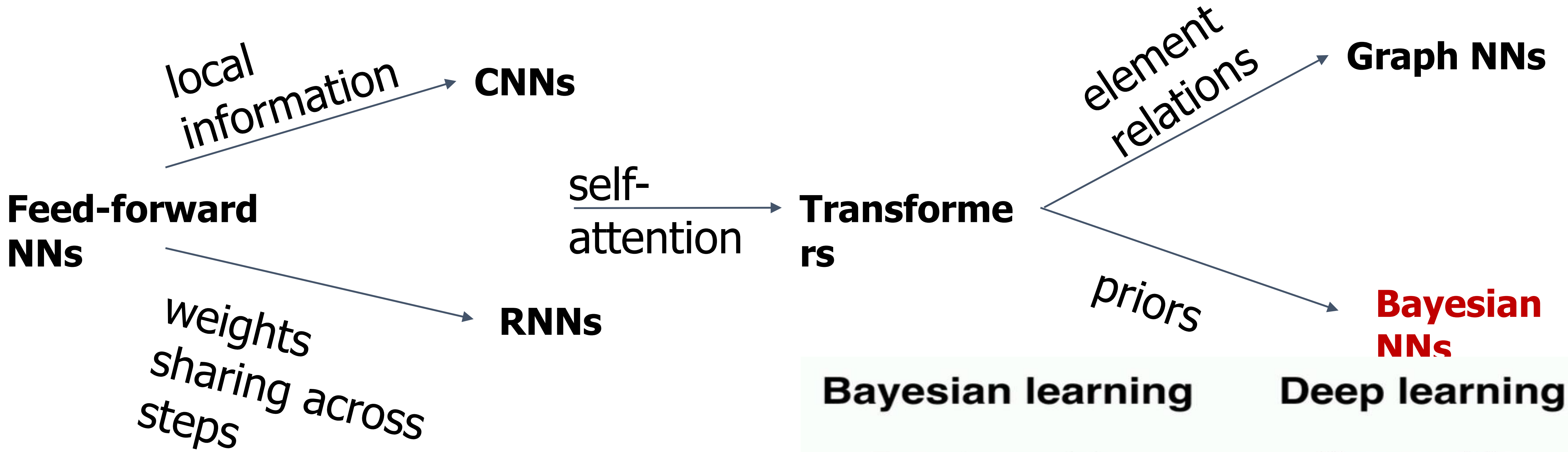


Graphs are natural representations for:

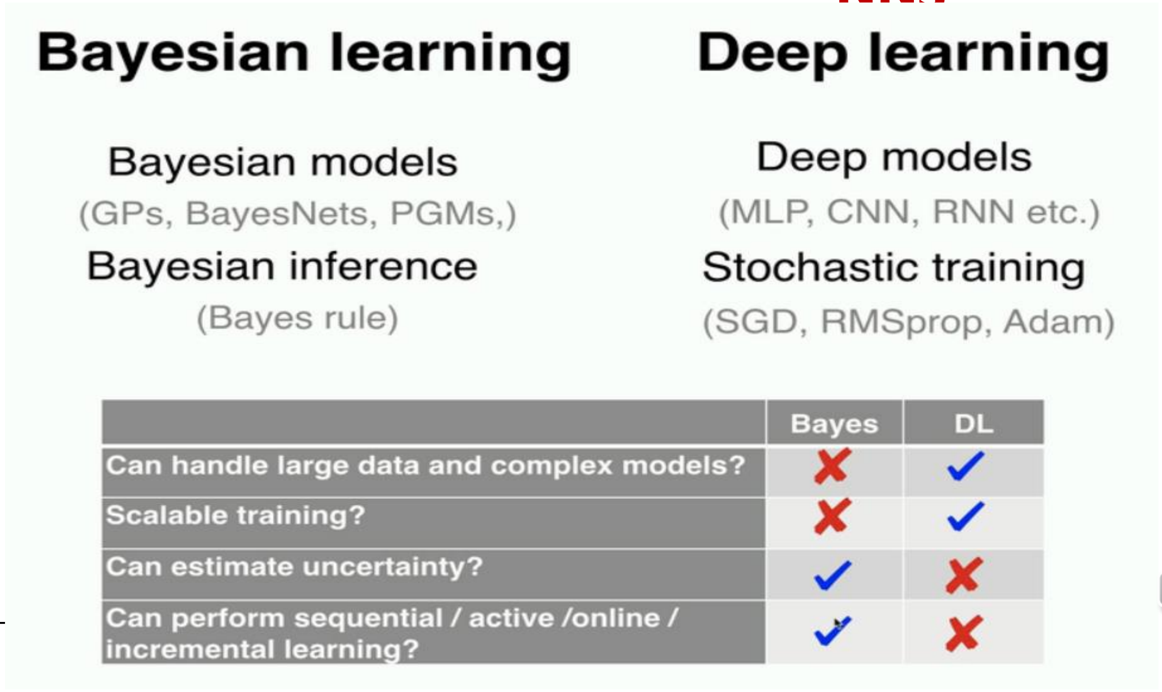
- inputs: social networks, knowledge bases, game states
- outputs: any joint distribution can be represented as a factor graph

Graph Convolutional Networks (Thomas Kipf, 2016)

Neural architecture evolution



Use a probability distribution over the network weights and output an average prediction of all sets of weights in that distribution



Bayesian learning	Deep learning
Bayesian models (GPs, BayesNets, PGMs,)	Deep models (MLP, CNN, RNN etc.)
Bayesian inference (Bayes rule)	Stochastic training (SGD, RMSprop, Adam)

	Bayes	DL
Can handle large data and complex models?	✗	✓
Scalable training?	✗	✓
Can estimate uncertainty?	✓	✗
Can perform sequential / active /online / incremental learning?	✓	✗

Deep Learning with Bayesian Principles (Emtiyaz Khan, NeurIPS 2019)

Architecture evolution

- Architectures come and go
  - LSTM-RNNs: largely replaced by Transformers for text, still used for time series (frequency trading)

- Be solution-focused, not architecture/buzzword-focused

Model selection: baselines

- **Random baseline**
  - Predict at random: uniform/following same distribution
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - E.g. autocompletion: predict the character most likely to appear based on a trigram lookup table
- **Existing solutions**:
  - Existing APIs

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following same distribution
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - E.g. autocompletion: predict the most character based on a trigram lookup table
- **Existing solutions**:
  - Existing APIs

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Uniform random	0.5	0.02
Distribution random	0.98	0.01
Most common	0.99	NaN

Model selection: baselines

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

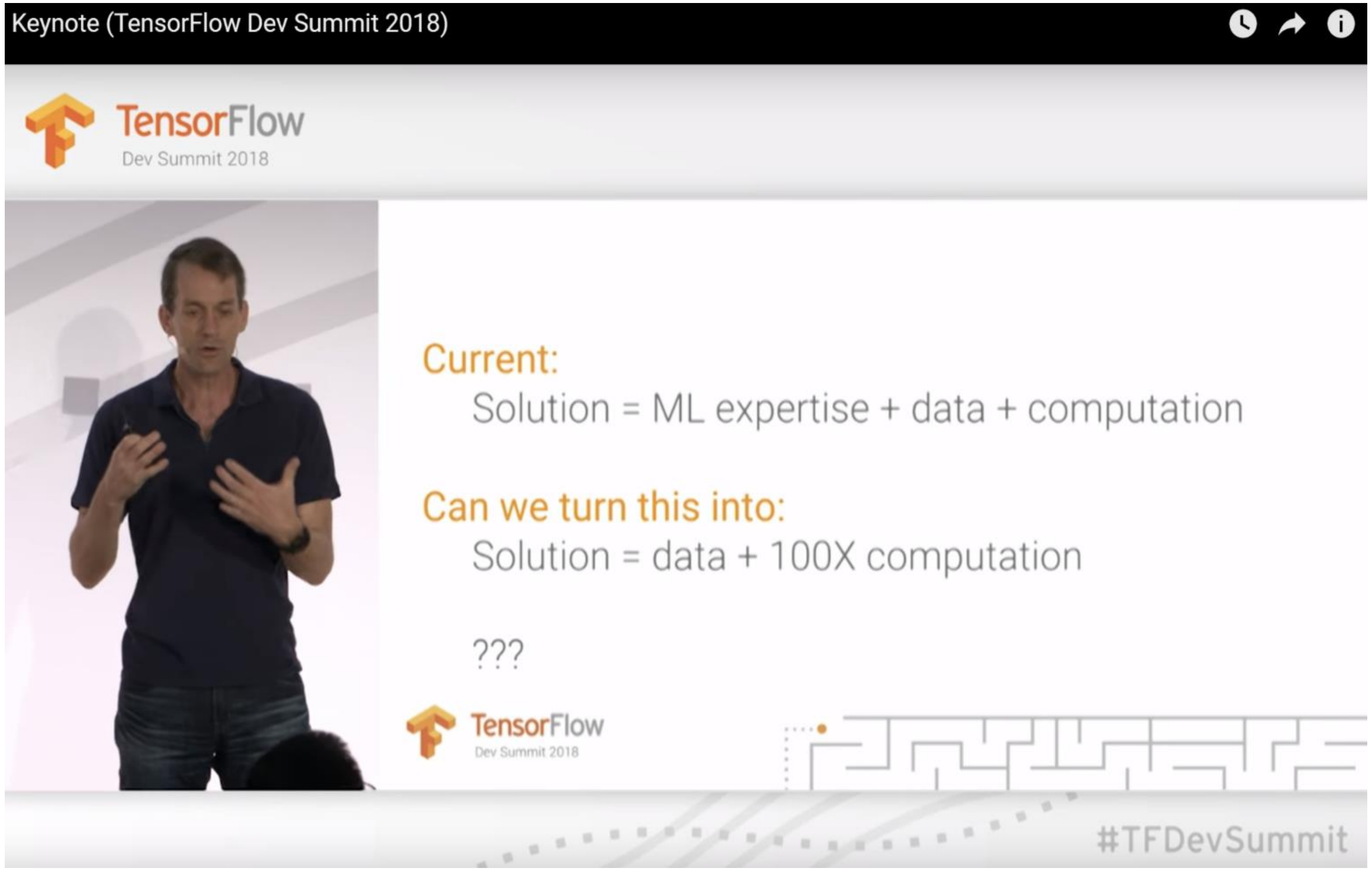
- **Random baseline**
  - Predict at random:
    - uniform
    - following same distribution
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - E.g. autocompletion: predict the most character based on a trigram lookup table
- **Existing solutions**:
  - Existing APIs

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Uniform random	0.5	0.02
Distribution random	0.98	0.01
Most common	0.99	NaN

AutoML

- A good ML researcher is someone who will automate themselves out of job!



Keynote (TensorFlow Dev Summit 2018)

TensorFlow Dev Summit 2018

Current:
Solution = ML expertise + data + computation

Can we turn this into:
Solution = data + 100X computation

???

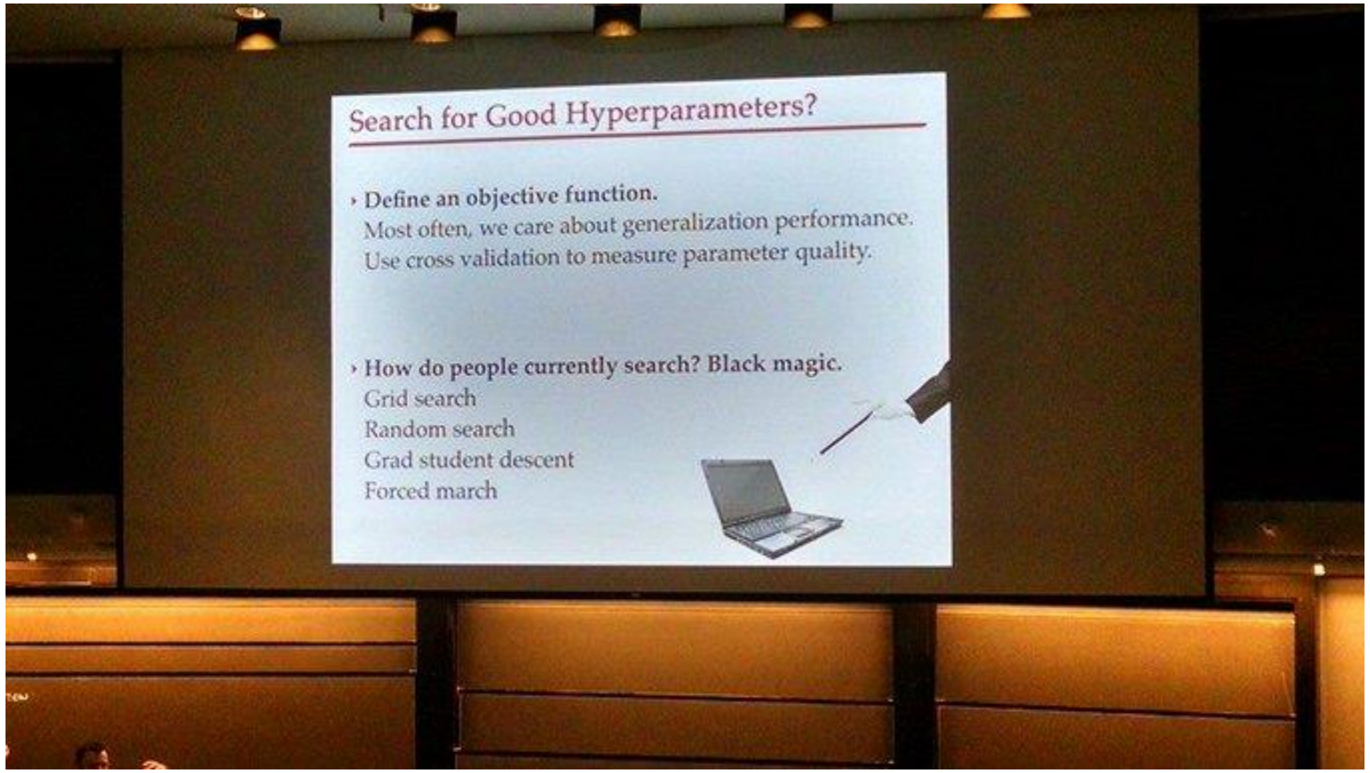
#TFDevSummit

AutoML

- Soft AutoML:
    - hyperparameter tuning
- Hard AutoML
    - neural architecture search
    - learned optimizer

Soft AutoML: Hyperparameter tuning

- Weaker models with well-tuned hyperparameters can outperform stronger, fancier models
  - On the State of the Art of Evaluation in Neural Language Models (Melis et al. 2018)



Search for Good Hyperparameters?

- Define an objective function.
  Most often, we care about generalization performance.
  Use cross validation to measure parameter quality.
- How do people currently search? Black magic.
  Grid search
  Random search
  Grad student descent
  Forced march

https://twitter.com/GuyZys/status/592847074170896384/photo/1

Soft AutoML: Hyperparameter tuning

"ResNet strikes back: An improved training procedure in timm" R. Wightman, H. Touvron, H. Jégou

- ResNets remain the gold-standard architecture in numerous scientific publications. They typically serve as the default architecture in studies, or as baselines when new architectures are proposed.

- Yet there has been significant progress on best practices for training neural networks since the inception of the ResNet architecture in 2015.

- Novel optimization & data-augmentation have increased the effectiveness of the training recipes.

Soft AutoML: Hyperparameter tuning

“ResNet strikes back: An improved training procedure in timm” R. Wightman, H. Touvron, H. Jégou

- In this paper, we re-evaluate the performance of the vanilla ResNet-50 when trained with a procedure that integrates such advances.
- We share competitive training settings and pre-trained models in the timm open-source library*, with the hope that they will serve as better baselines for future work.
- For instance, with our more demanding training setting, a vanilla ResNet-50 reaches 80.4% top-1 accuracy at resolution 224x224 on ImageNet-val without extra data or distillation.

*Py**T**orch **Im**age **M**odels (timm) is a collection of image models, layers, utilities, optimizers, schedulers, data-loaders / augmentations, and reference training / validation scripts that aim to pull together a wide variety of SOTA models with ability to reproduce ImageNet training results.

Hyper-Parameter Optimization (HPO)

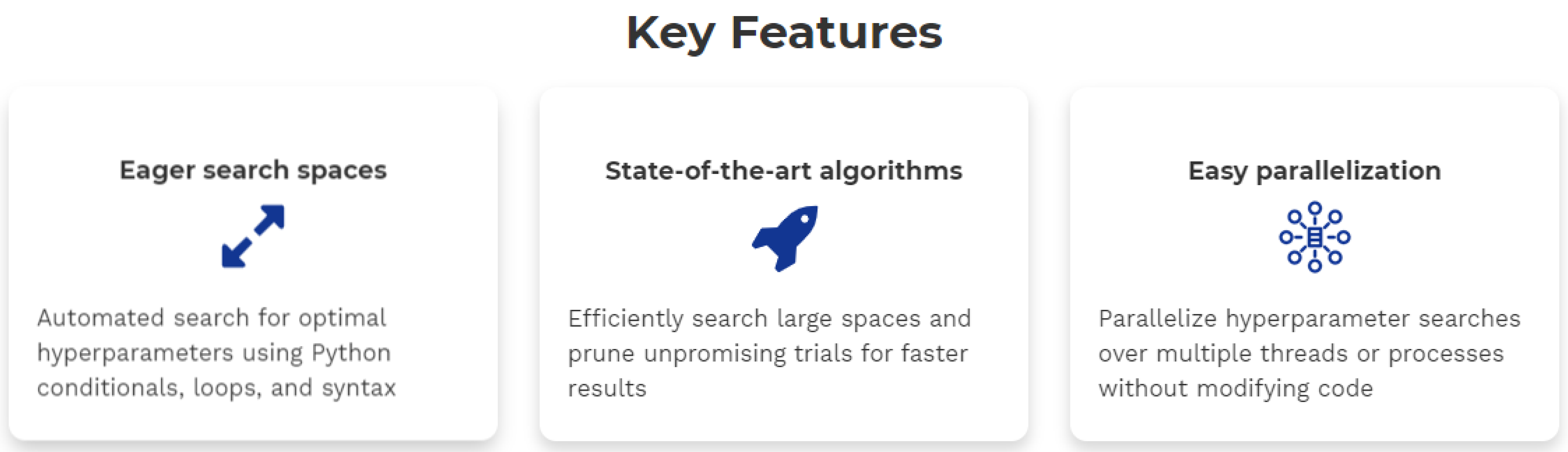
- Selecting the best hyper-parameters
- HPO methods are based on training the model several times
- Each new hyper-parameter results in adding a new dimension and exponentially increases the computational complexity. Hence, HPO requires high resource use
- Ideally HPO needs to take the target platform into account. When optimizing for embedded systems and mobile devices, energy consumption and memory limitations (hardware-aware ML) and model accuracy and hardware efficiency need to be optimized in conjunction.

Hyper-Parameter Optimization (HPO)

Optuna: Optimize Your Optimization

An open-source hyper-parameter optimization framework.

Aims to automate hyperparameter search.



Key Features

Eager search spaces

Automated search for optimal hyperparameters using Python conditionals, loops, and syntax

State-of-the-art algorithms

Efficiently search large spaces and prune unpromising trials for faster results

Easy parallelization

Parallelize hyperparameter searches over multiple threads or processes without modifying code

https://optuna.org/

NAS: Neural architecture search

- **Search space**
  - Set of operations
    - e.g. convolution, fully-connected, pooling
  - How operations can be connected

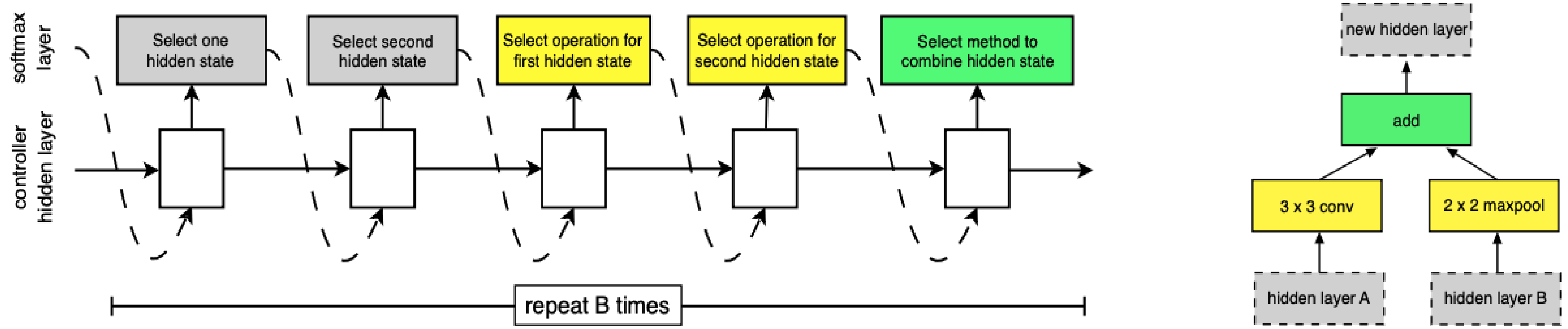


Figure 3. Controller model architecture for recursively constructing one block of a convolutional cell. Each block requires selecting 5 discrete parameters, each of which corresponds to the output of a softmax layer. Example constructed block shown on right. A convolutional cell contains B blocks, hence the controller contains $5B$ softmax layers for predicting the architecture of a convolutional cell. In our experiments, the number of blocks B is 5.

Learning Transferable Architectures for Scalable Image Recognition (Zoph et al., 2017)

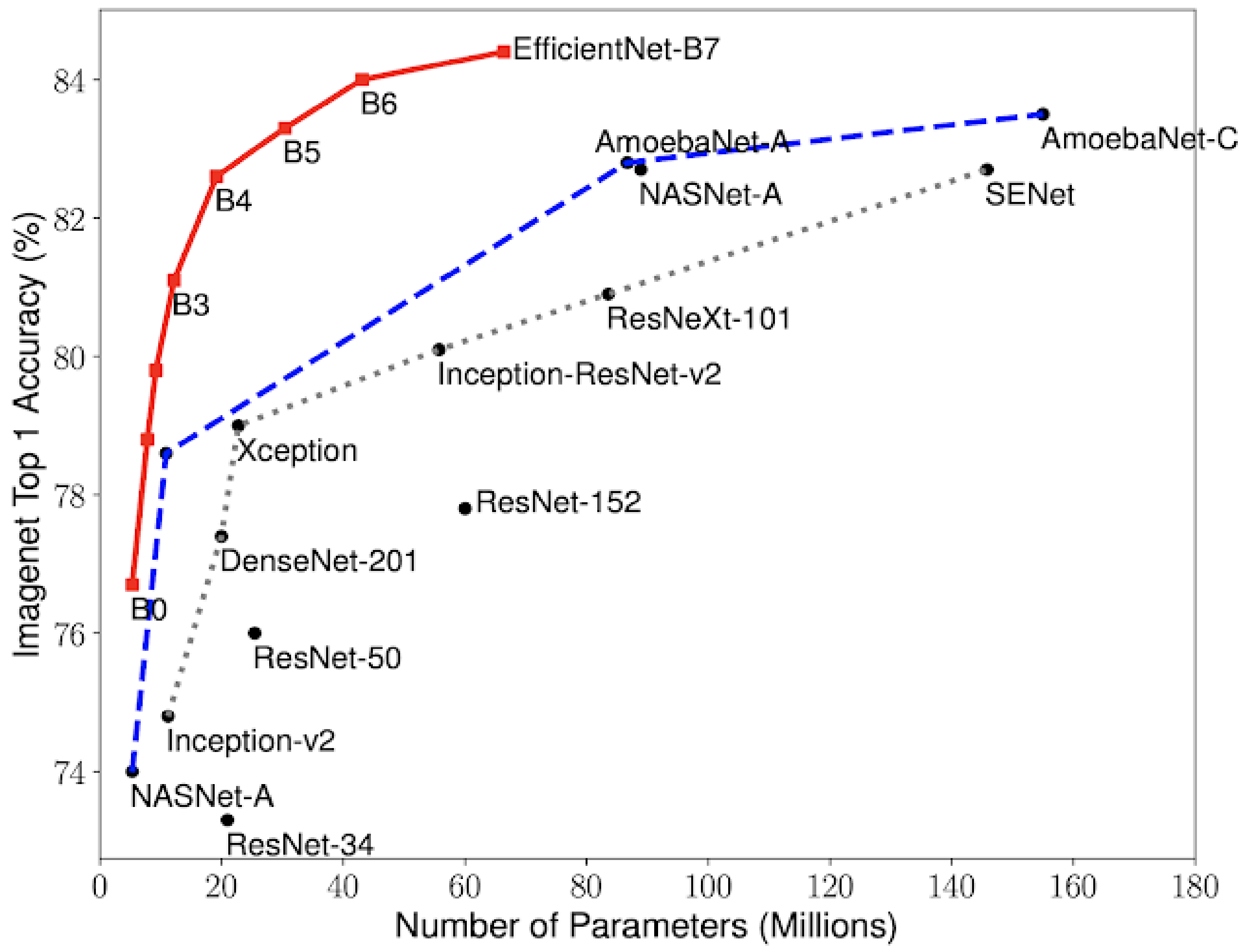
NAS: Neural architecture search

- **Search space**
- **Performance estimation strategy**
  - How to evaluate many candidate architectures?
  - Ideal: should be done without having to re-construct or re-train them from scratch.

NAS: Neural architecture search

- **Search space**
- **Performance estimation strategy**
- **Search strategy**
  - Random
  - Reinforcement learning
    - reward the choices that improve performance estimation
  - Evolution
    - mutate an architecture
    - choose the best-performing offsprings
    - so on

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks



A systematical study of model scaling to carefully balancing network depth, width, and resolution can lead to better performance

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (Tan et Le., 2019)

Learning: architecture + learning algorithm

- We have two main components: Architecture itself and learning algorithm used to train the architecture.
- Learning algorithm is a set of functions that specifies how to update the weights. Also called **optimizers:** Adam, Momentum, SGD, …
- We are coming up with new architecture, can we also improve the learning algorithm?

Meta Learning: Learning to learn

Meta learning applies automatic learning algorithms to metadata about machine learning experiments.

The main goal is to use such metadata to understand how automatic learning can become flexible in solving learning problems, hence to improve the performance of existing learning algorithms or to learn (induce) the learning algorithm itself.

Learned optimizer

"… the models created with these techniques can outperform humans on a large number of tasks. But what's interesting is these machine learning techniques are still designed by humans and this is a bottleneck because we're sort of limited by the creativity of about what we can design in these algorithms.

So a motivating factor for me, for my research for the last couple of years has been trying to use machine learning techniques to improve machine learning"

Luke Metz, Google Brain

Data augmentation: goals

- Improve model's performance overall or on certain classes
- Generalize better
- Enforce certain behaviors

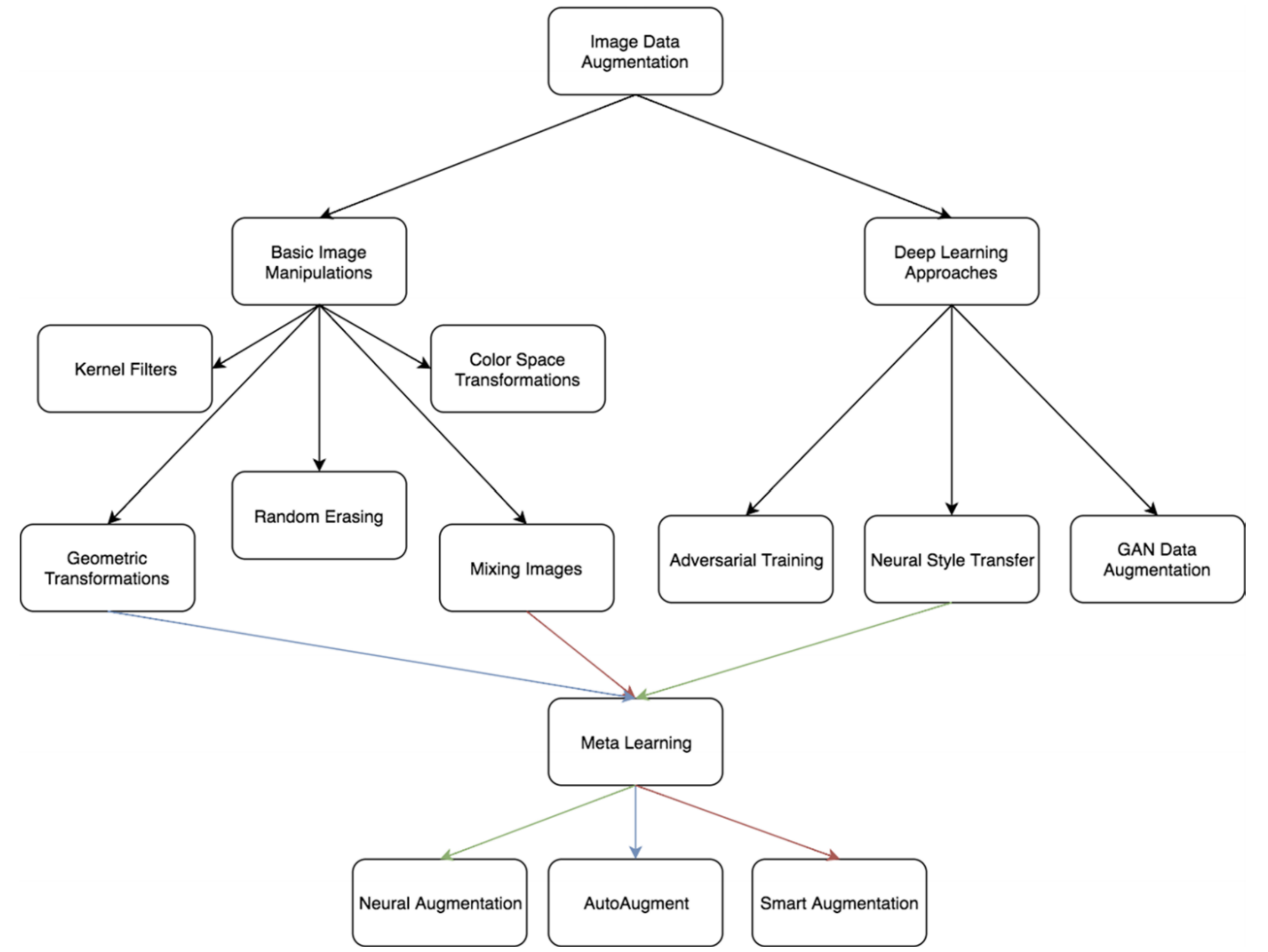
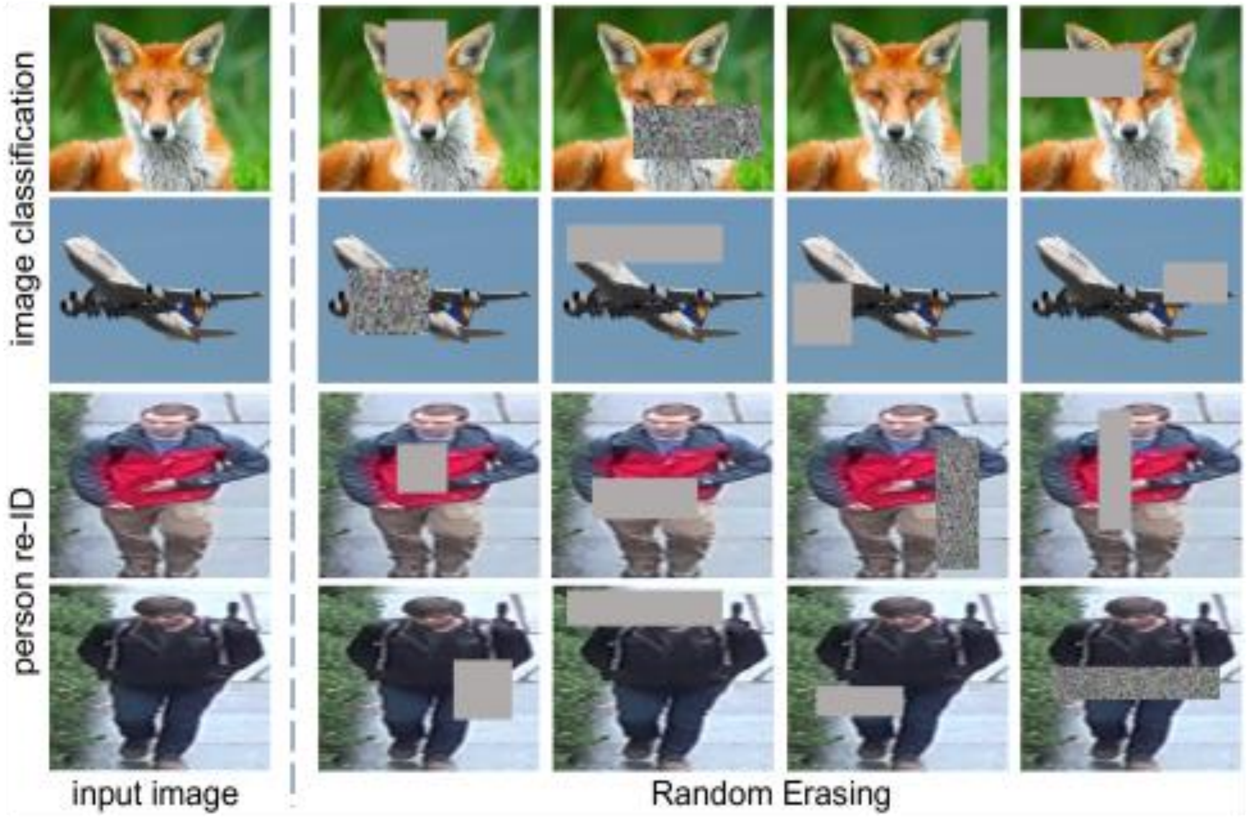


Fig. 2 A taxonomy of image data augmentations covered; the colored lines in the figure depict which data augmentation method the corresponding meta-learning scheme uses, for example, meta-learning using Neural Style Transfer is covered in neural augmentation [36]

Data augmentation: image

- Artificially enlarge the dataset using label-preserving image manipulations
  - random cropping, flipping, erasing



Random Erasing Data Augmentation (Zhong et al., 2017)

mixup

- Create convex combination of samples of different classes
  - Original: image 1 - label 3 (cat), image 2 - label 4 (dog). All labels are discrete
  - Mixup: 70% dog, 30% cat - label 3.3
- Incentivize models to learn linear relationships
  - Assumption: linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples

mixup: Beyond Empirical Risk Minimization (Zhang et al., 2017)

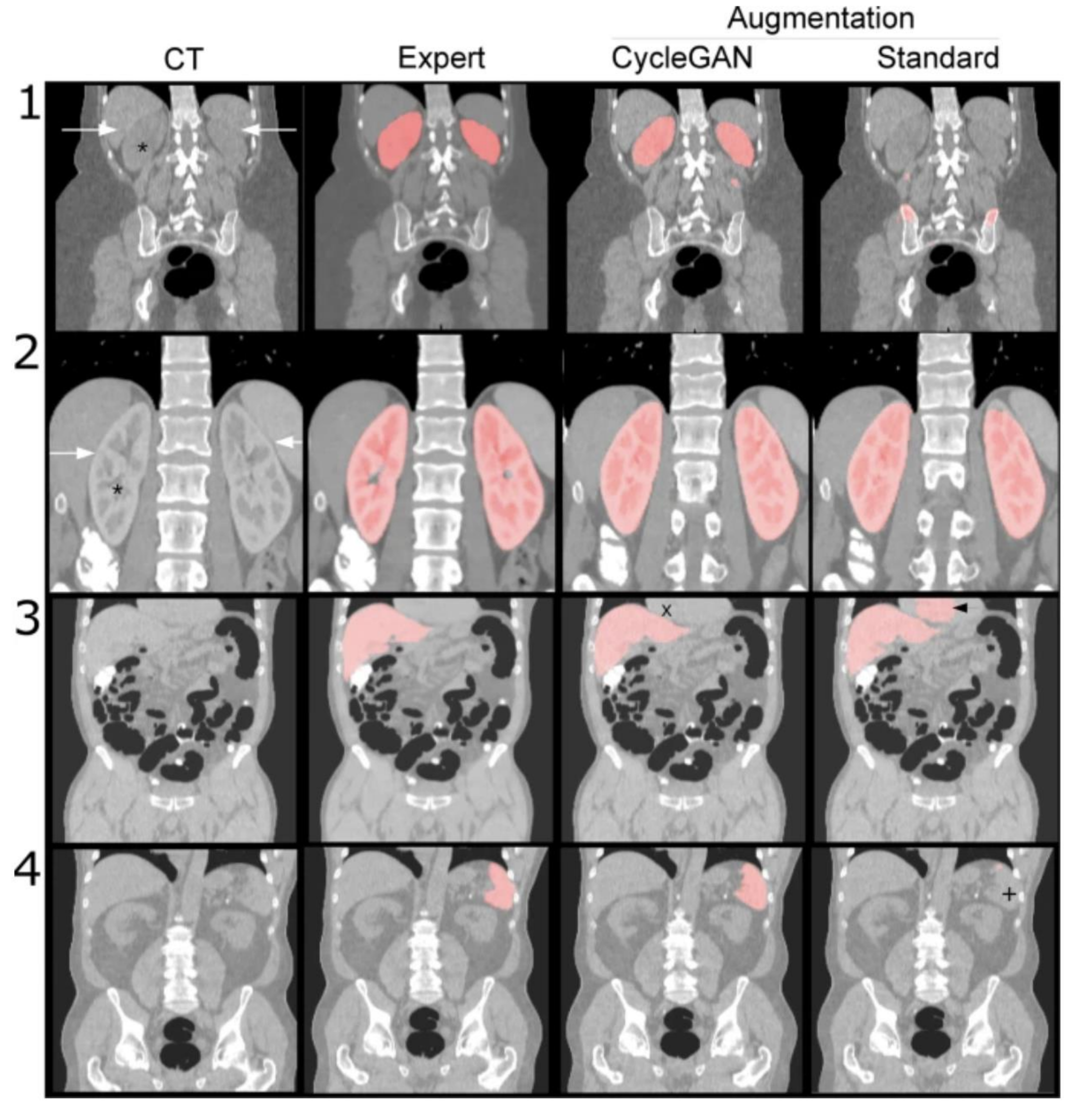
mixup

- Improves generalization on speech and tabular data
- Can be used to stabilize the training of GANs

https://forums.fast.ai/t/mixup-data-augmentation/22764

Data augmentation: GAN

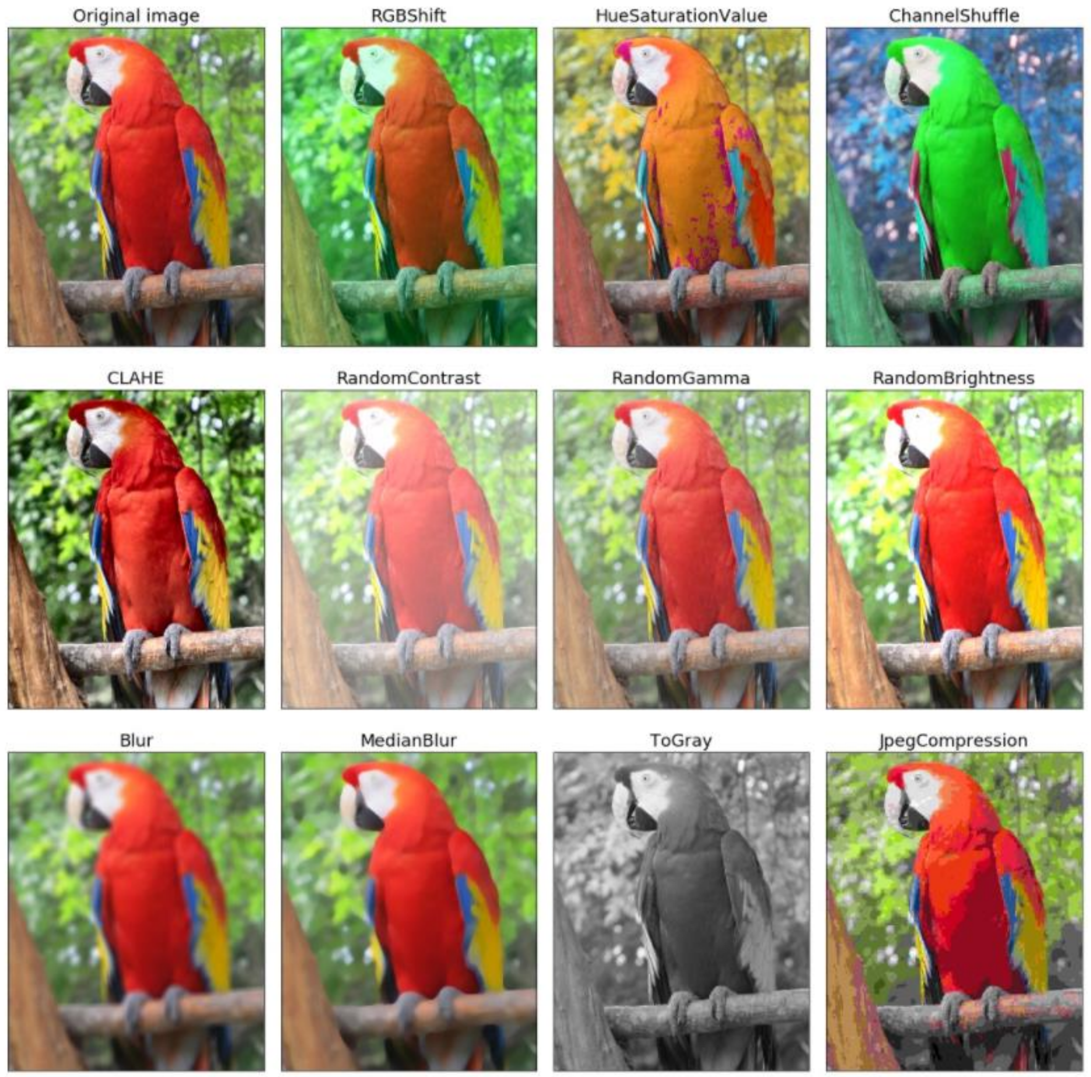
Example: kidney segmentation with data augmentation by CycleGAN



Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks (Sandfort et al., 2019)

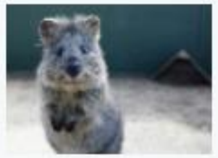
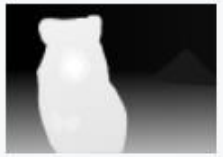
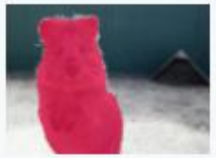
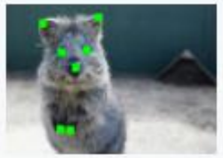
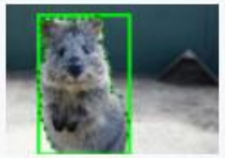
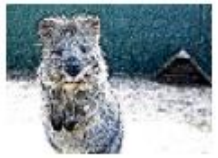
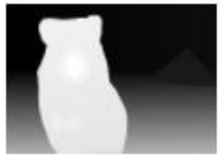
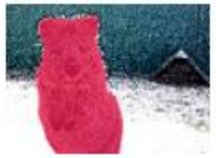
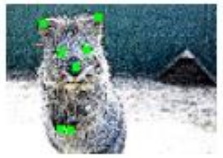
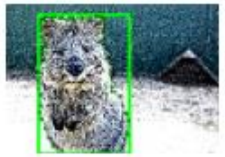
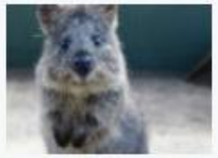
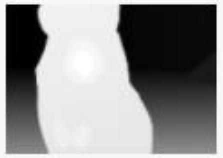
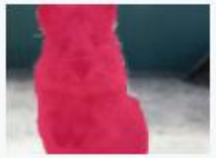
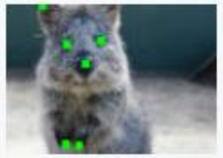
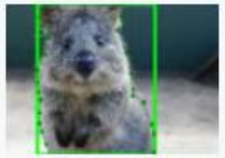
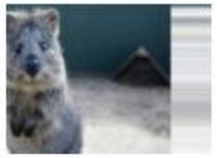
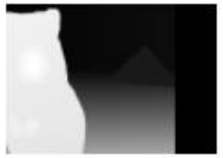
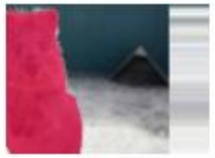
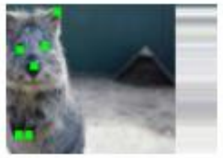
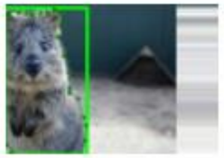
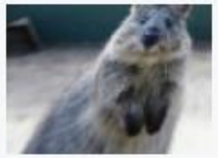
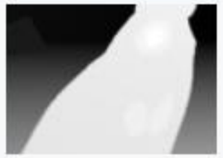
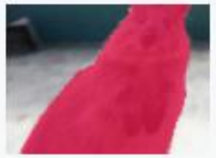
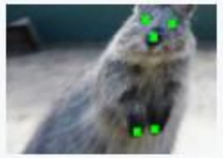
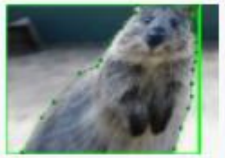
Albumentations

- https://github.com/albumentations-team/albumentations



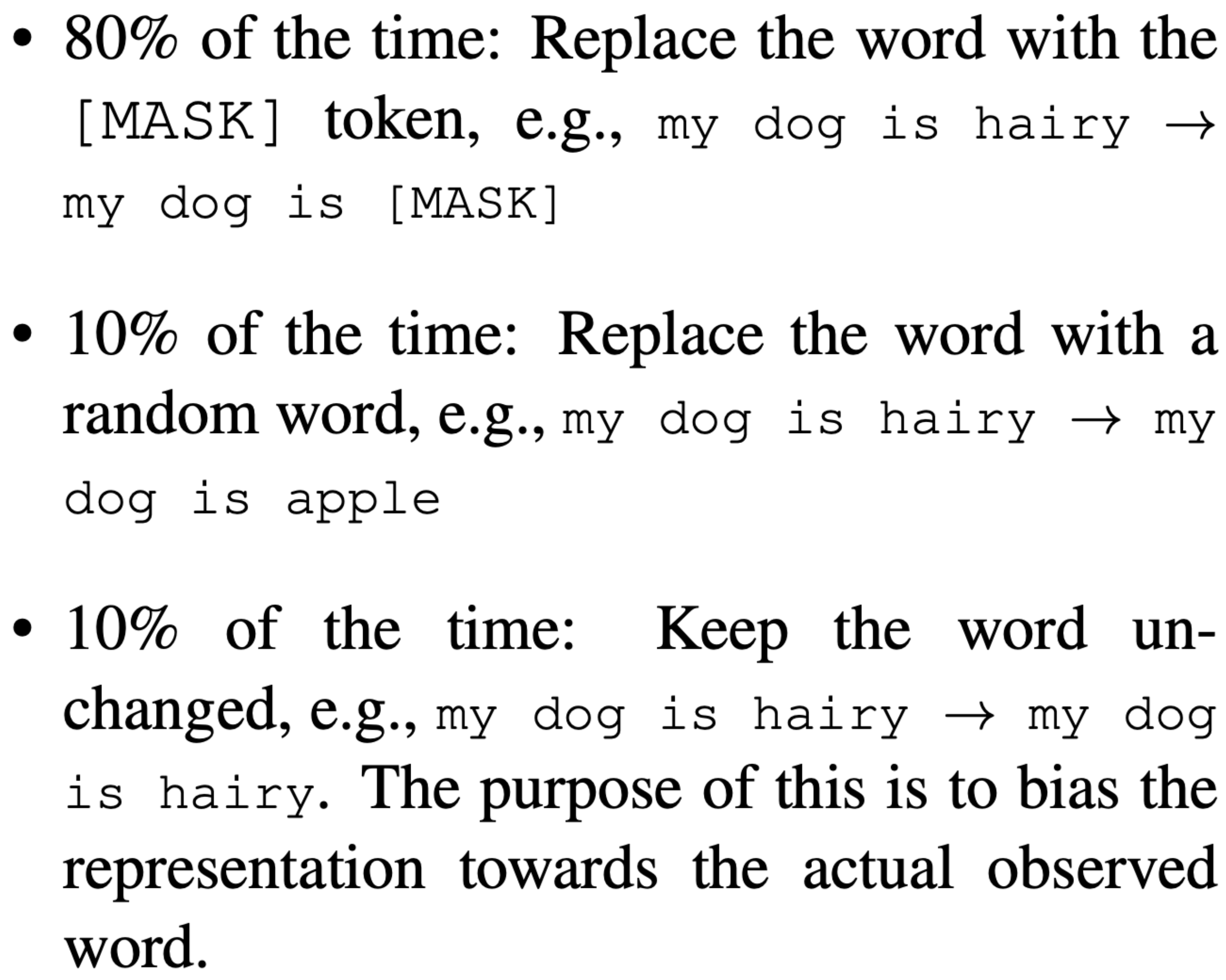
Imgaug

- https://github.com/aleju/imgaug

	Image	Heatmaps	Seg. Maps	Keypoints	Bounding Boxes, Polygons
Original Input					
Gauss. Noise + Contrast + Sharpen					
Affine					
Crop + Pad					
Fliplr + Perspective					

Data augmentation: text

- Random replacement
  - e.g. BERT (10% * 15% = 1.5%)



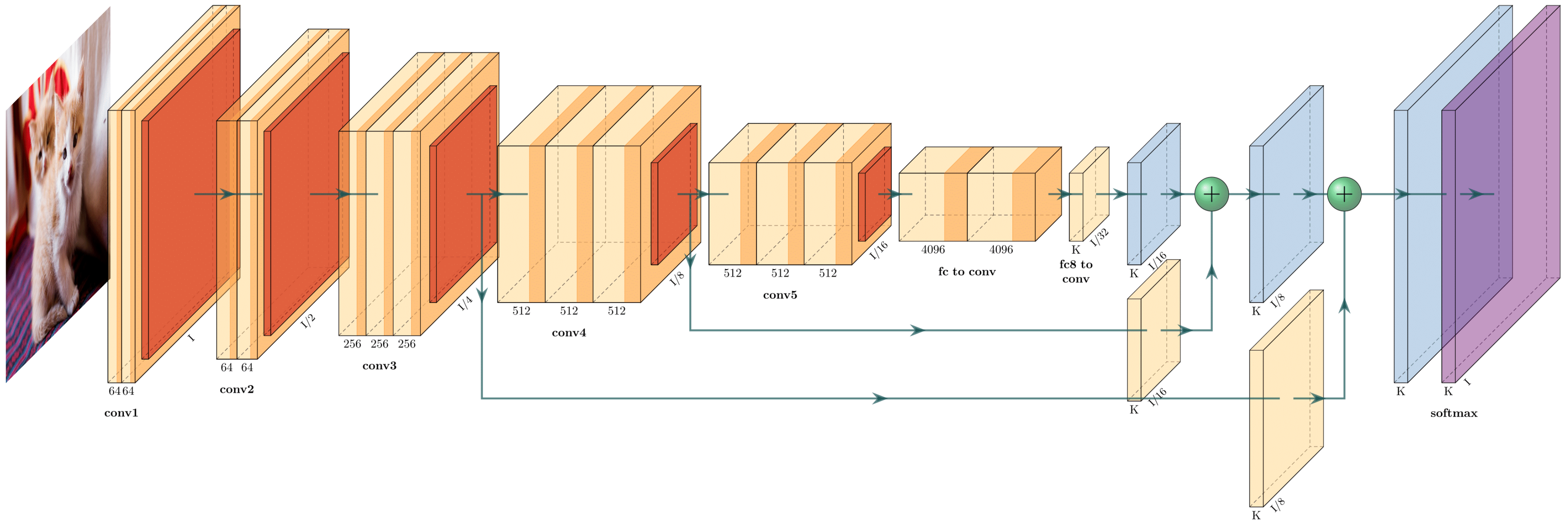
- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., 2018)

Drawing a Network Diagram

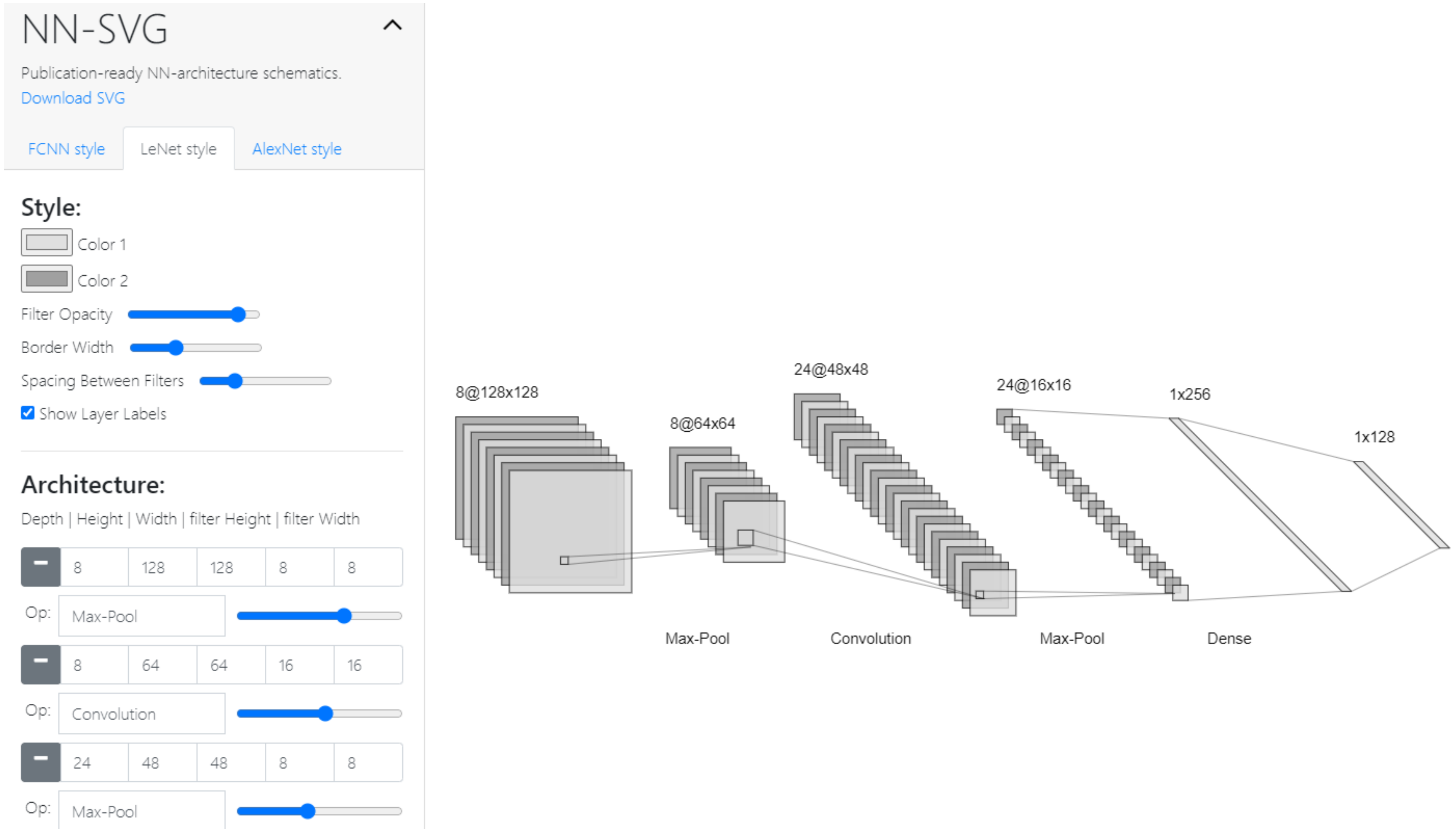
Latex code for drawing neural networks for reports and presentation

https://github.com/HarisIqbal88/PlotNeuralNet



Drawing a Network Diagram – NN SVG

- https://alexlenail.me/NN-SVG/



NN-SVG

Publication-ready NN-architecture schematics.

Download SVG

FCNN style | LeNet style | AlexNet style

Style:

Color 1

Color 2

Filter Opacity

Border Width

Spacing Between Filters

☑ Show Layer Labels

Architecture:

Depth | Height | Width | filter Height | filter Width

8	128	128	8	8

Op: Max-Pool

8	64	64	16	16

Op: Convolution

24	48	48	8	8

Op: Max-Pool