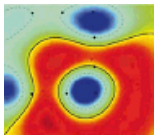
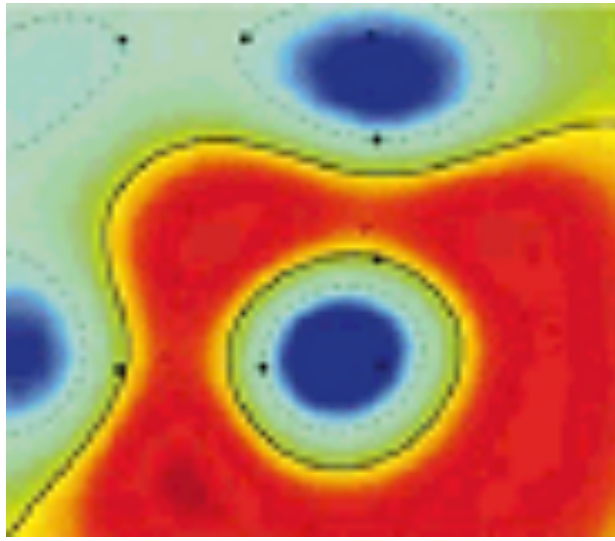


WiSe 2024/25

Deep Learning 1

Technische Universität Berlin



Lecture 6

Overfitting & Robustness (2)

Outline

Worst-Case Analysis

- The problem of adversarial examples
- Adversarial robustness

Adding Predictive Uncertainty

- Why predictive uncertainty?
- Density networks
- Ensemble models

Adding Prior Knowledge

- Translation invariance, local smoothness, etc.
- Feature reuse (transfer / multitask / self-supervised learning)

Part 1

Worst-Case Analysis

Motivations

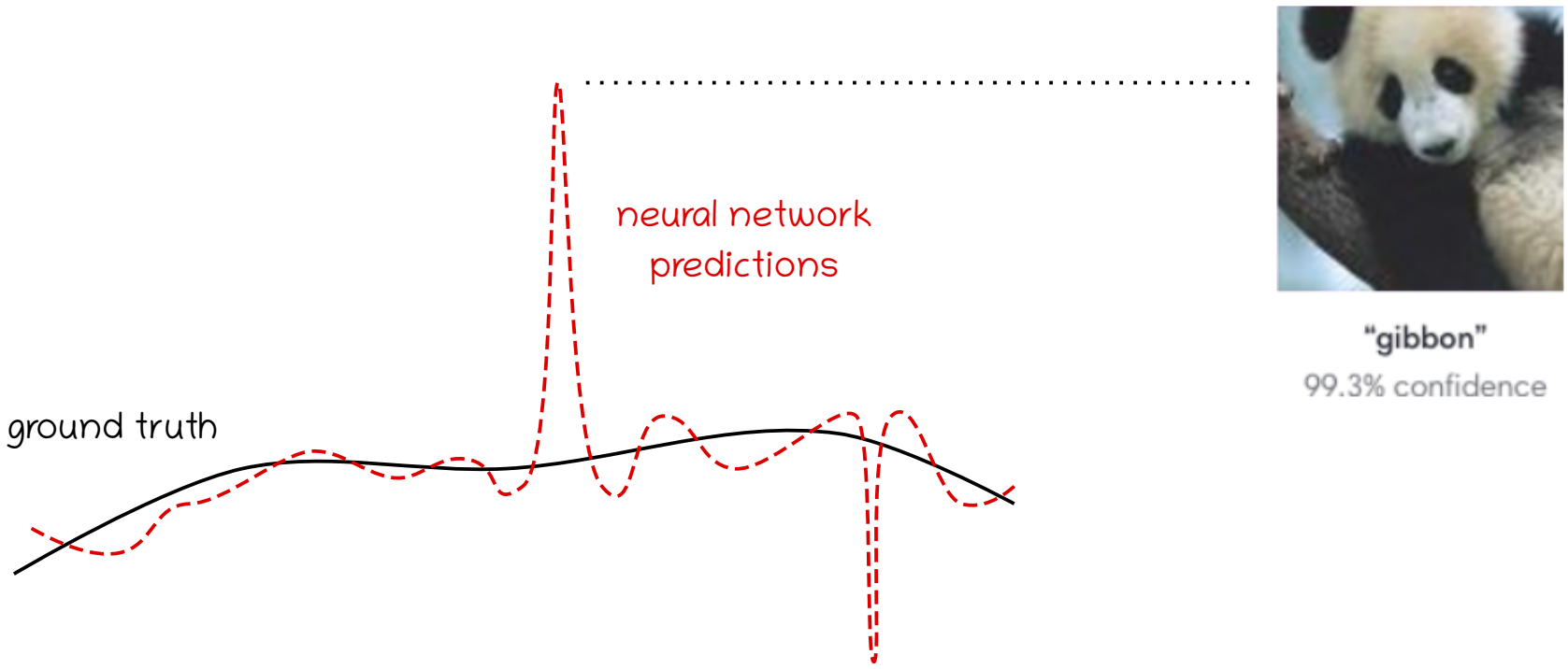
Risk aversion

- One big error can often be more harmful than many small errors, e.g. a system being controlled by a neural network may be tolerant to small errors (which can be corrected subsequently), but not to a big error from which one cannot recover.

Adversarial components

- Even though the neural network may perform well on average, an adversary may craft inputs that steer the ML system towards the worst-case decision behavior.

Worst-Case Analysis



Typical causes of large errors:

- High dimensionality of input space allows to finely craft patterns to which the network responds highly.
- High depth/nonlinearity implies that the function is locally steeper than necessary.

Example: Adversarial Examples

- Carefully crafted nearly invisible perturbations of an existing data point can cause the prediction of a neural network to change drastically, while leaving almost no trace of the attack.

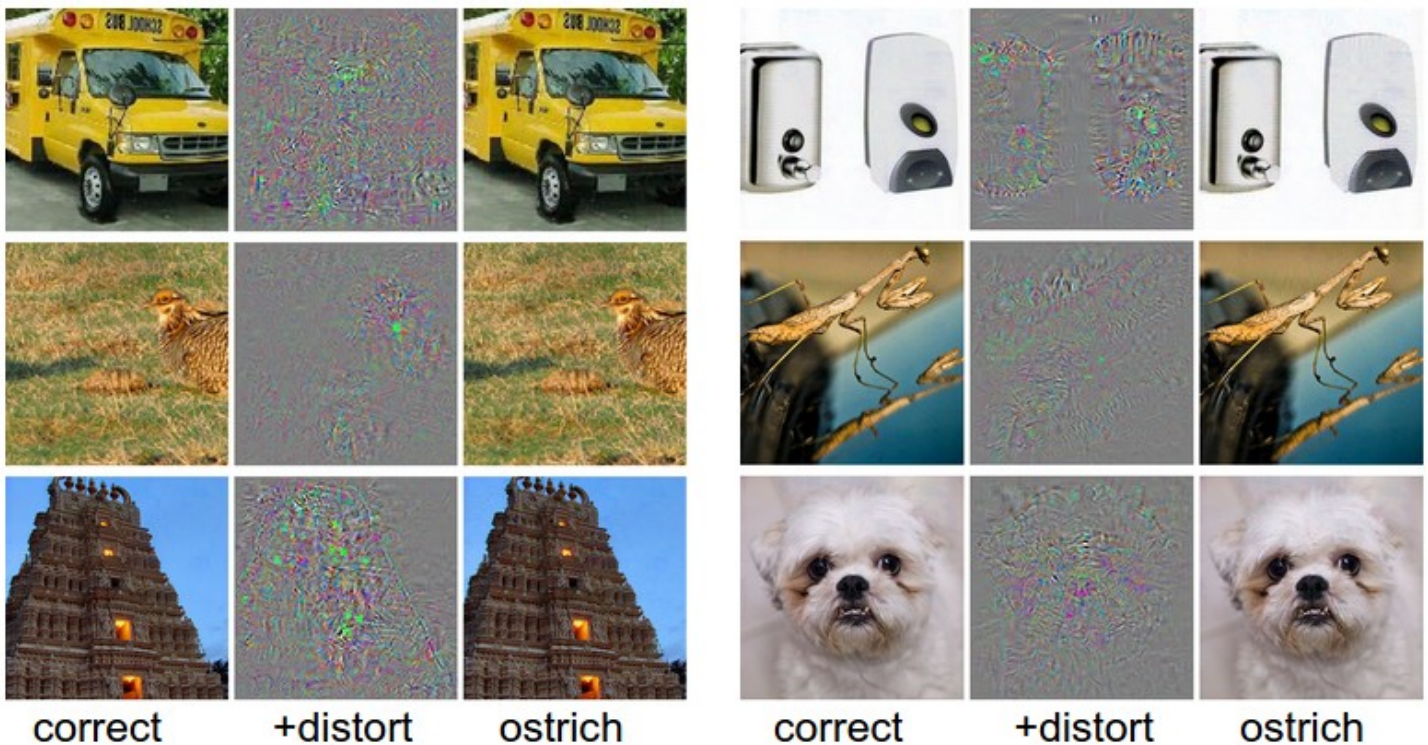


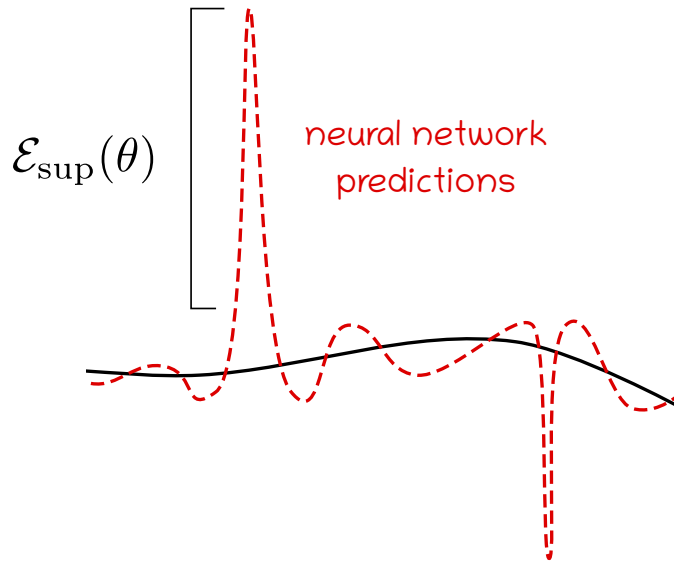
Image source: https://arxiv.org/abs/1312.6199

- Serious concern in various applications (e.g. biometric identification, reading traffic signs).

Addressing Worst-Case Behavior

Enhanced Regularization:

- Search for high local variations of the decision function and add these variations as a term of the error function to minimize.
- In practice, this can take the form of generating adversarial examples, and forcing them to be predicted in the same way as the original data.
- More generic approaches based on Lipschitz-continuity (e.g. spectral norm regularization) can also be used.



Data Preprocessing:

- In practice, one can also address worst-case behavior by applying dimensionality reduction (e.g. blurring images) before applying the neural network.

Part 2

Predictive Uncertainty

Predictive Uncertainty

Practical motivations:

- Understand when we can trust the model in a more precise way than just looking at the overall error.
- Enables the user to be prompted when the model is unsure, in which case, the user can decide e.g. to collect more data, or to perform the prediction manually.

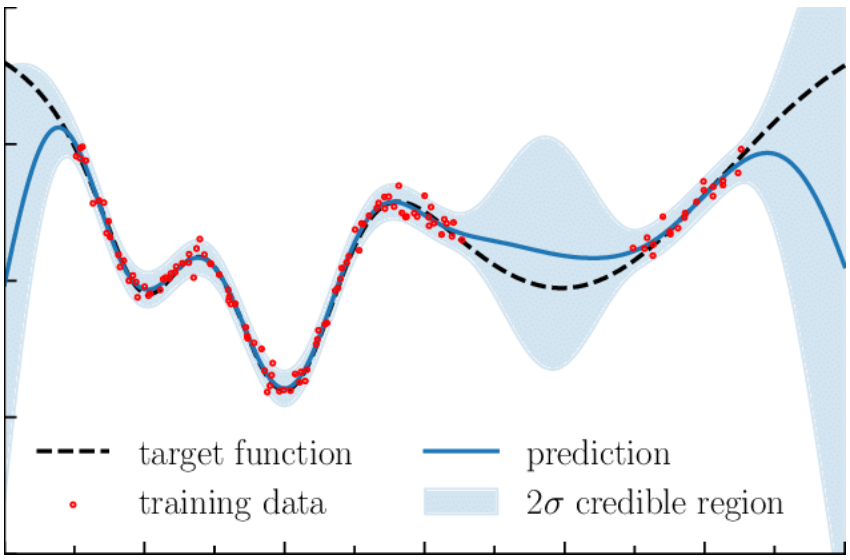


Image source: https://doi.org/10.1103/PhysRevD.98.063511

Predictive Uncertainty

Approach 1:

- Explicitly encoding the uncertainty estimate in the neural network, i.e. have one output neuron for predicting the actual value of interest, and a second output neuron for predicting the uncertainty associated to this prediction.
- For example, one predicts that the output is distributed according to the random variable $y \sim \mathcal{N}(\mu, \sigma^2)$ where μ and σ are the two neural network outputs. (How to train these models will be presented in Lecture 7.)

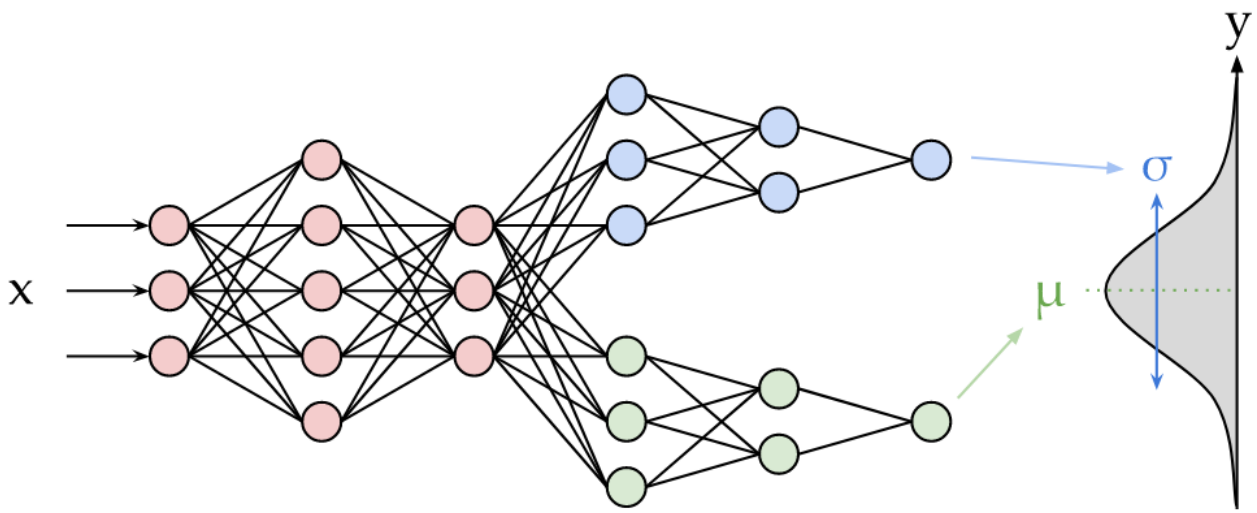


Image source: https://brendanhasz.github.io/2019/07/23/bayesian-density-net.html

Predictive Uncertainty

Approach 2:

- Train an ensemble of neural networks and measure prediction uncertainty as the discrepancy of predictions of each network in the ensemble, e.g. for an ensemble of n neural networks with respective outputs $o_1, \ldots, o_n$, we generate the two aggregated outputs

$$\mu = \mathrm{avg}(o_1, \ldots, o_n) \quad \text{and} \quad \sigma = \mathrm{std}(o_1, \ldots, o_n).$$

that represent the final prediction and its uncertainty.

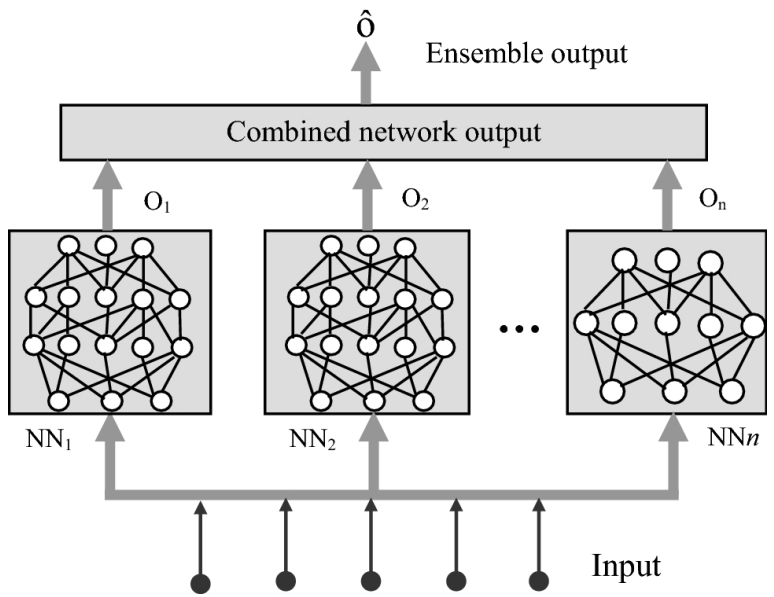


Image source: https://doi.org/10.1007/s00521-019-04359-7

Predictive Uncertainty

Approach 2 (cont.):

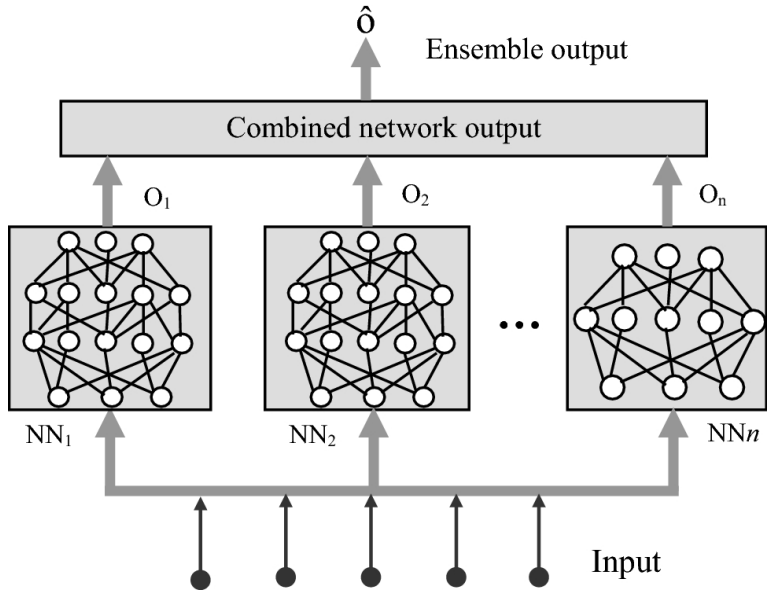


Image source: https://doi.org/10.1007/s00521-019-04359-7

- Each network in the ensemble may have a different initialization, may receive different input features, and may be trained on different subsets of data.
- Uncertainty can then be understood as the effect of neural network random initialization, feature selection, data sample.
- The more heterogeneous the ensemble, the higher the estimate of uncertainty.

Part 3

Beyond Regularization: Prior Knowledge

Prior Knowledge

Recap:

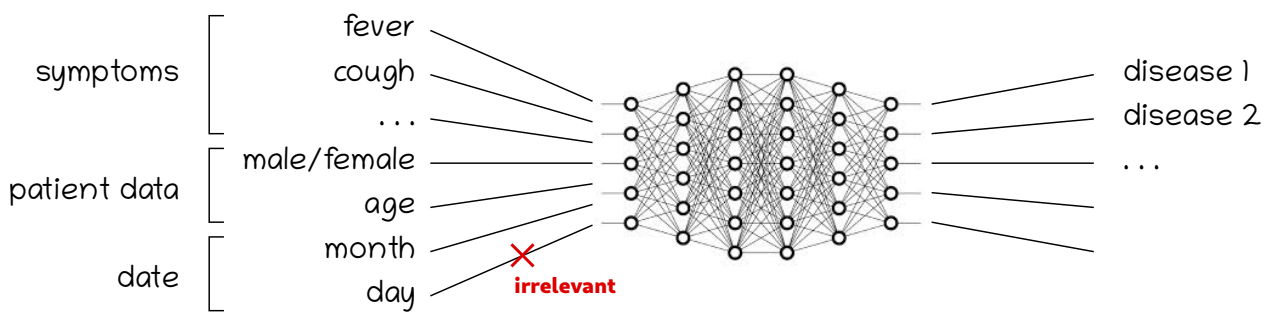
- Machine learning is affected by data quality issues (e.g. scarcity of data or labels, spurious correlations in the dataset, under-representation of some part of the distribution, shift between data available for training and data when deployed).

Idea:

- **There is no point to learn from the data what we already know.** What we already know (our prior knowledge) should ideally be hard-coded into the model.

Example:

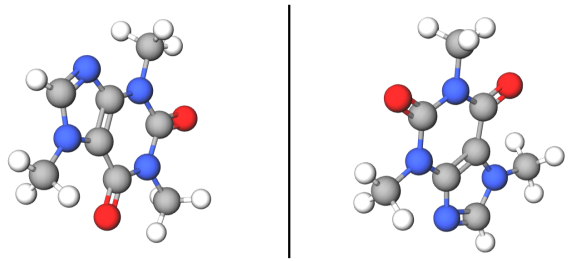
- In specific tasks, certain features are known to have no effect on the quantity to predict. It is better to not give them as input to the neural network.



Physical Invariances

Example: Rotation/Translation Invariance

- Rotating or translating a molecule maintains its atomization energy constant.



- Rotation invariance can be ensured e.g. by encoding the molecule by the pairwise distances between its atoms rather than its 3d coordinates, and feeding these distances to a neural network.

$$\Phi(\mathbf{r}) = \begin{pmatrix} \|\mathbf{r}_1 - \mathbf{r}_2\|, & \|\mathbf{r}_1 - \mathbf{r}_3\|, & \dots & \|\mathbf{r}_1 - \mathbf{r}_L\| \\ & \|\mathbf{r}_2 - \mathbf{r}_3\|, & & \|\mathbf{r}_2 - \mathbf{r}_L\| \\ & & \ddots & \vdots \\ & & & \|\mathbf{r}_{L-1} - \mathbf{r}_L\| \end{pmatrix}$$

Physical Invariances

Example: Modeling interaction between two molecules

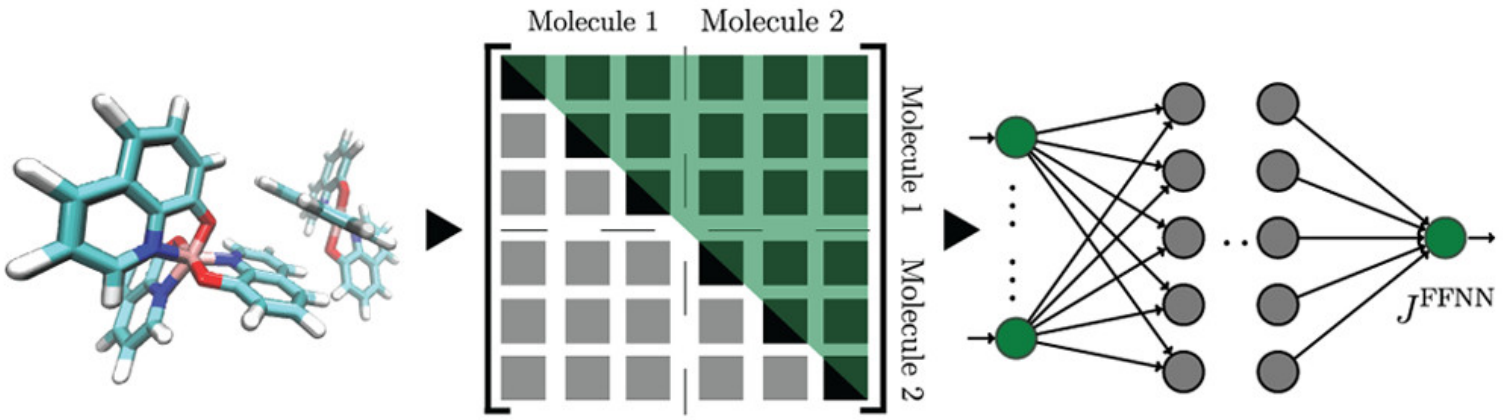


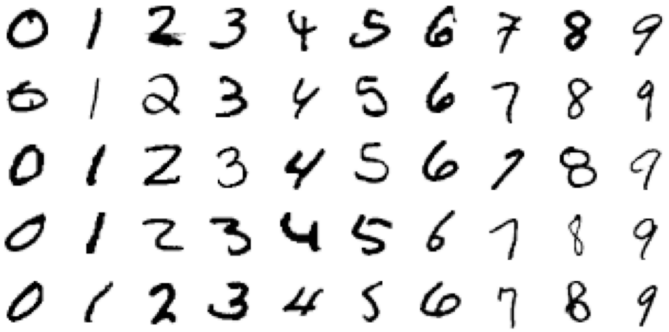
Image source: https://doi.org/10.1021/acs.jctc.8b01285

- Distances are fed as input to a plain neural network.
- Work as long as atom of the molecules can be indexed (i.e. approach stops working when the molecules received as input are of arbitrary shape and size).

Soft Invariances

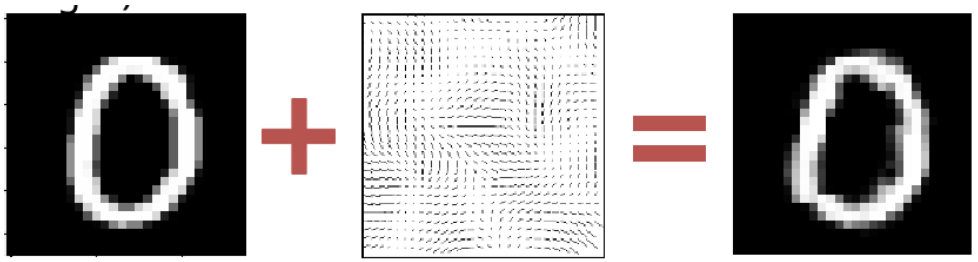
Example: Handwritten digits recognition

- Rotating digits by a few degrees usually does not change class membership.
- Some exceptions, e.g. rotating a '1' may transform it into a '7'.



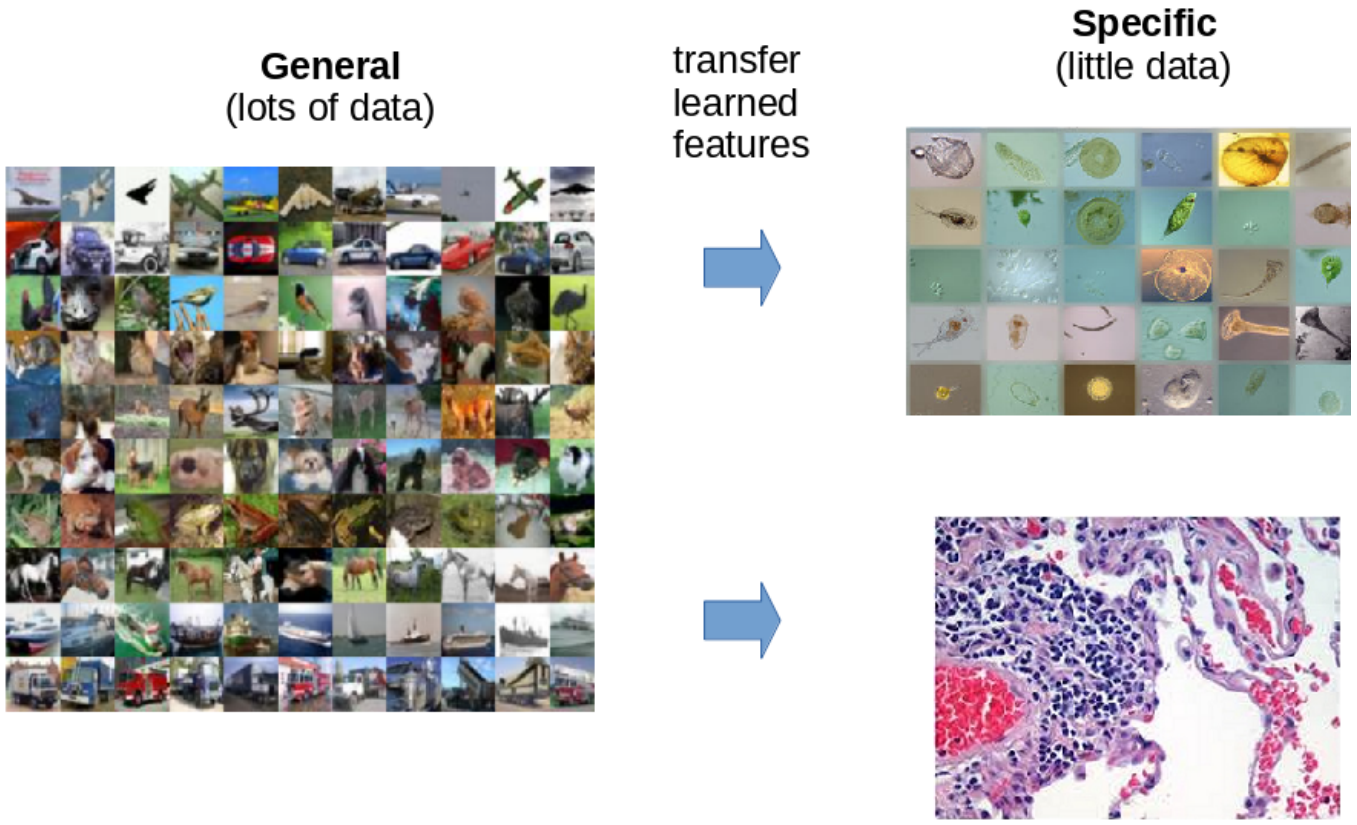
Approaches to build invariance:

- Use purposely designed neural network architectures. E.g. scattering networks, pooling networks, etc.
- Augment the dataset with elastic distortions, and train the neural network on the extended dataset.



Source: M. Lerousseau "From Papers to Github #1: A practical guide to handwritten digits classifier & dataset preprocessing in Python and tensorflow"
https://marvinler.github.io/2017/03/11/from-papers-to-github-1.html

Feature Reuse / Transfer Learning



- Certain tasks have intrinsically little annotated data (e.g. scientific data), due to the cost of labeling by an expert.
- However, they have similarities with other tasks with much more data (e.g. general-purpose image recognition). In both cases, one needs to extract features such as edge or color detectors to solve the task.

Transfer Learning with Deep Networks

Approach:

- When two tasks are *related*, we can train a big neural network on the first task with abundant data, and reuse features in intermediate layer for the task of interest.

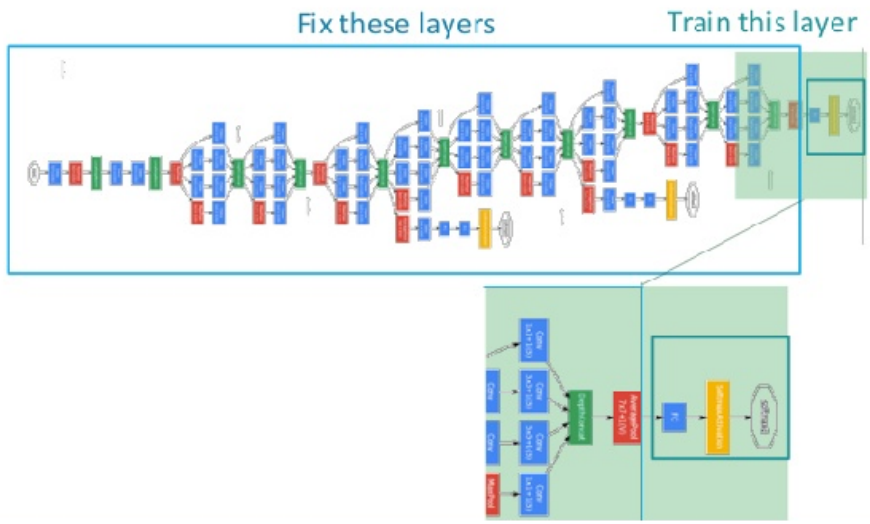


Image Source: Mark Chang, Applied Deep Learning 11/03 Convolutional Neural Networks

- This type of transfer learning is very common in applied research. For image recognition tasks, researchers typically start from a state-of-the-art network for vision (e.g. ResNet) pretrained on ImageNet, and retrain the top layers on the specific task.

How to Generate Useful Features

Classical approach:

- Learn a model to classify a more general task (e.g. image recognition).

Self-supervised learning approach

- Create an artificial task where labels can be produced automatically, and whose solution requires features that are needed for the task of interest (more in DL2).

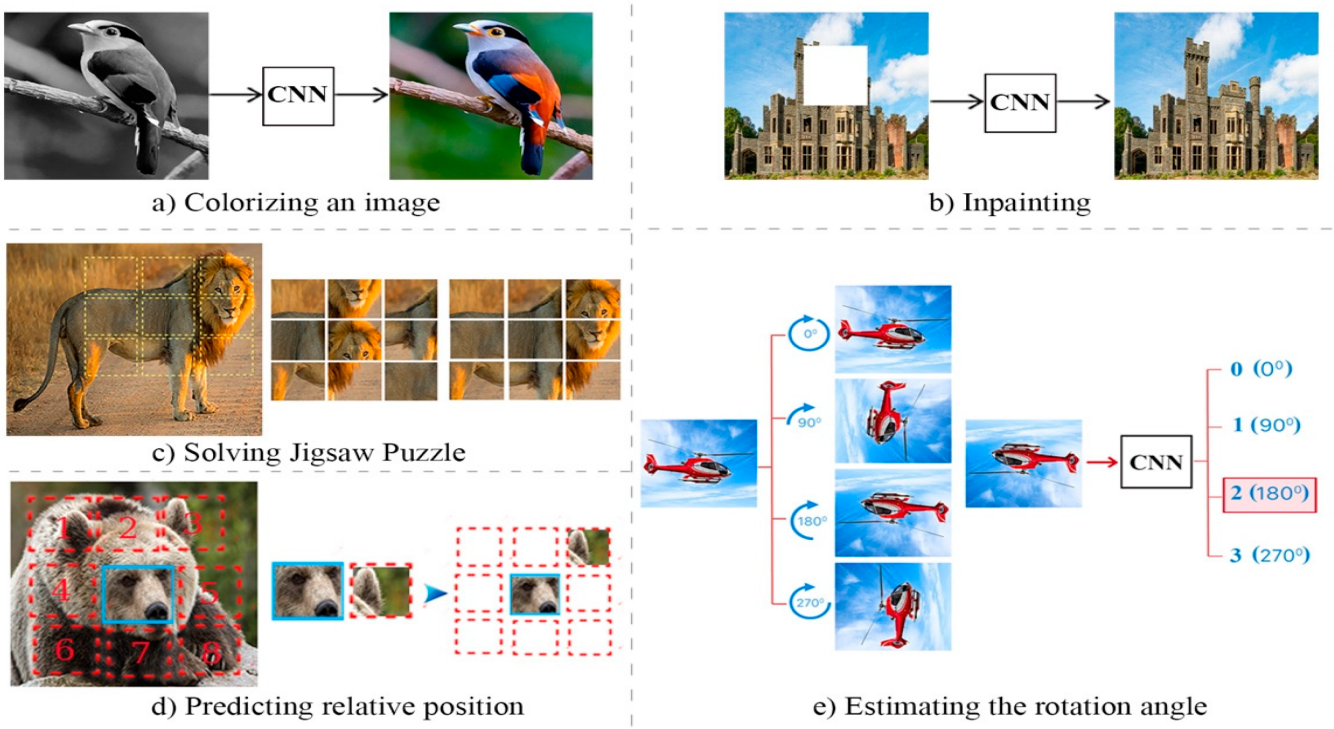


Image source: https://doi.org/10.3390/e24040551

Summary

Summary

- In practice, expected prediction accuracy may not be the most relevant quantity. The true practical usefulness of a system is often better characterized by its worst-case performance.
- It is often desirable to equip neural network models with some measure of predictive uncertainty so that the network can tell the user when to trust and when not to trust the prediction.
- There is no point to learn what we already know. Prior knowledge (e.g. invariances, shared features) can be introduced in neural networks. As a result, the model becomes less affected by overfitting and also more robust.