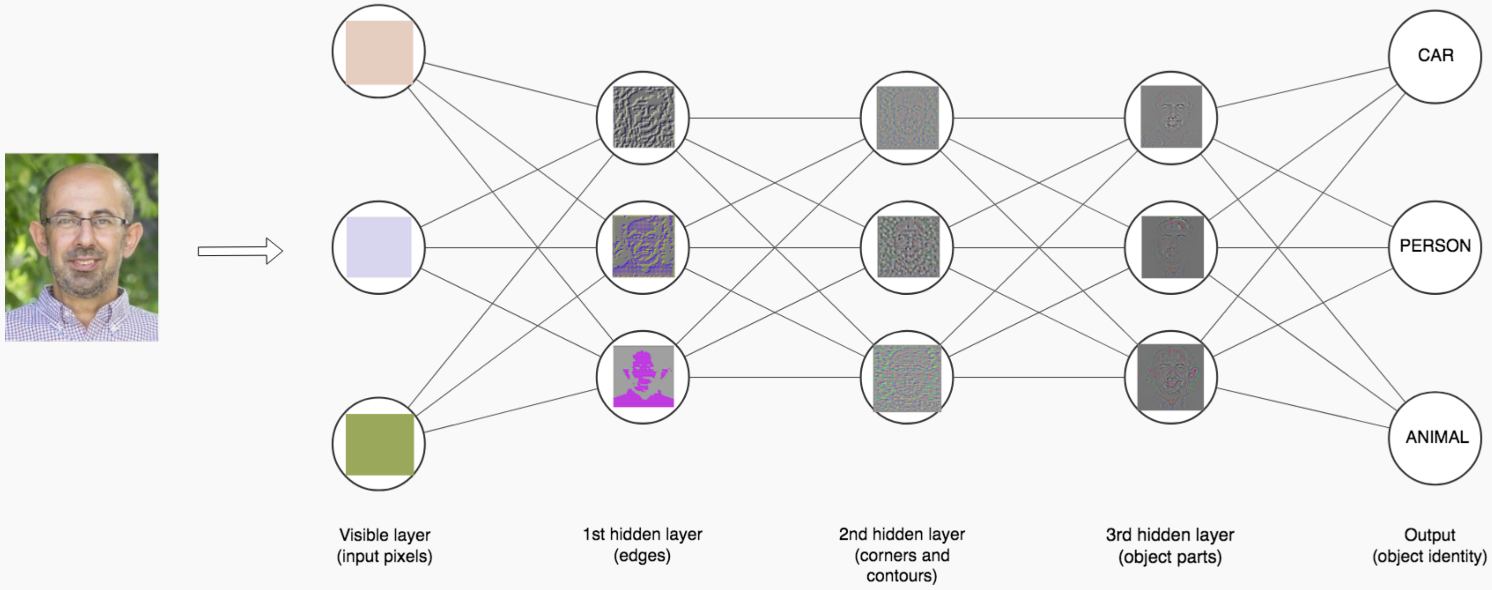
Lecture 33 : NN Review

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner

Review: neural networks

Deep learning is a mathematical framework for learning **layered representations** from data. Neural networks are models for learning such representations; they are structured in actual layers stacked next to each other.



When are neural networks useful?

Deep learning is not always the right tool for the job.

XGBoost, Random Forest or **Gradient Boosting** is usually a better choice for:

- Tabular data, data with engineered features

NNs shine on:

- Image recognition
- Natural language processing
- Autonomous driving
- Speech recognition

Do NNs benefit from feature engineering? Yes, when there is domain knowledge, and/or when there is too few training data.

Ingredients for training a neural network

INPUT

The input data is a 2D data matrix of shape `(batch_size, features)`.

All inputs to a NN need to be tensors. Values in the tensors should be small (in the [-1, 1] or [0, 1] range). Heterogeneous data should be normalized. Some feature engineering might benefot small datasets.

We pass the number of features as a parameter to the input layer. If our training data consists of 6000 observations, each with 4 features, then

```
input_shape = (4,)
```

Ingredients for training a neural network

INPUT

Categorical variables should be encoded using one of the options we have learned:

- **Integer Encoding**: each unique label is mapped to an integer;
- **One Hot Encoding**: each label is mapped to a binary vector;

To input images in a FFN we must `flatten` them first, that is, reshape the 2D or 3D array of pixel values to a 1D array (these will be the features)

Ingredients for training a neural network

LOSS FUNCTION

Regression: **Mean squared error** (`mean_squared_error`)

Classification: **Cross-entropy**

- For binary target variables we use `binary_crossentropy`.
- For multiclass target variables, integer encoded, we use `sparse_categorical_crossentropy`.
- For multiclass target variables, one-hot encoded, we use `categorical_crossentropy`.

Ingredients for training a neural network

OUTPUT

```
number of features = 4

model.add(tf.keras.layers.Dense(4, activation='softmax’))
```

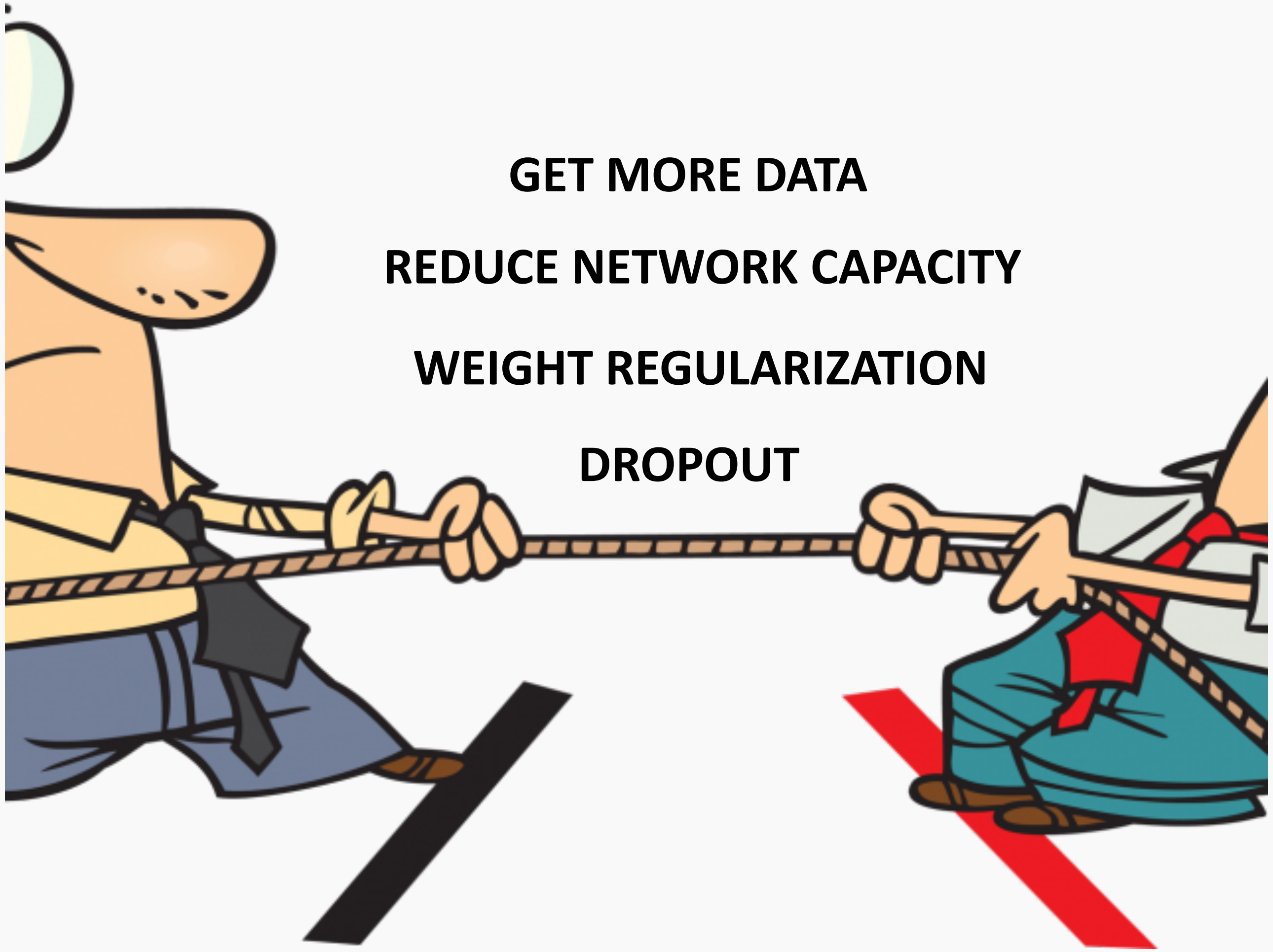
Design choices for neural networks

Design choices for neural networks

Unfortunately there is no magic formula

Optimization vs. Generalization

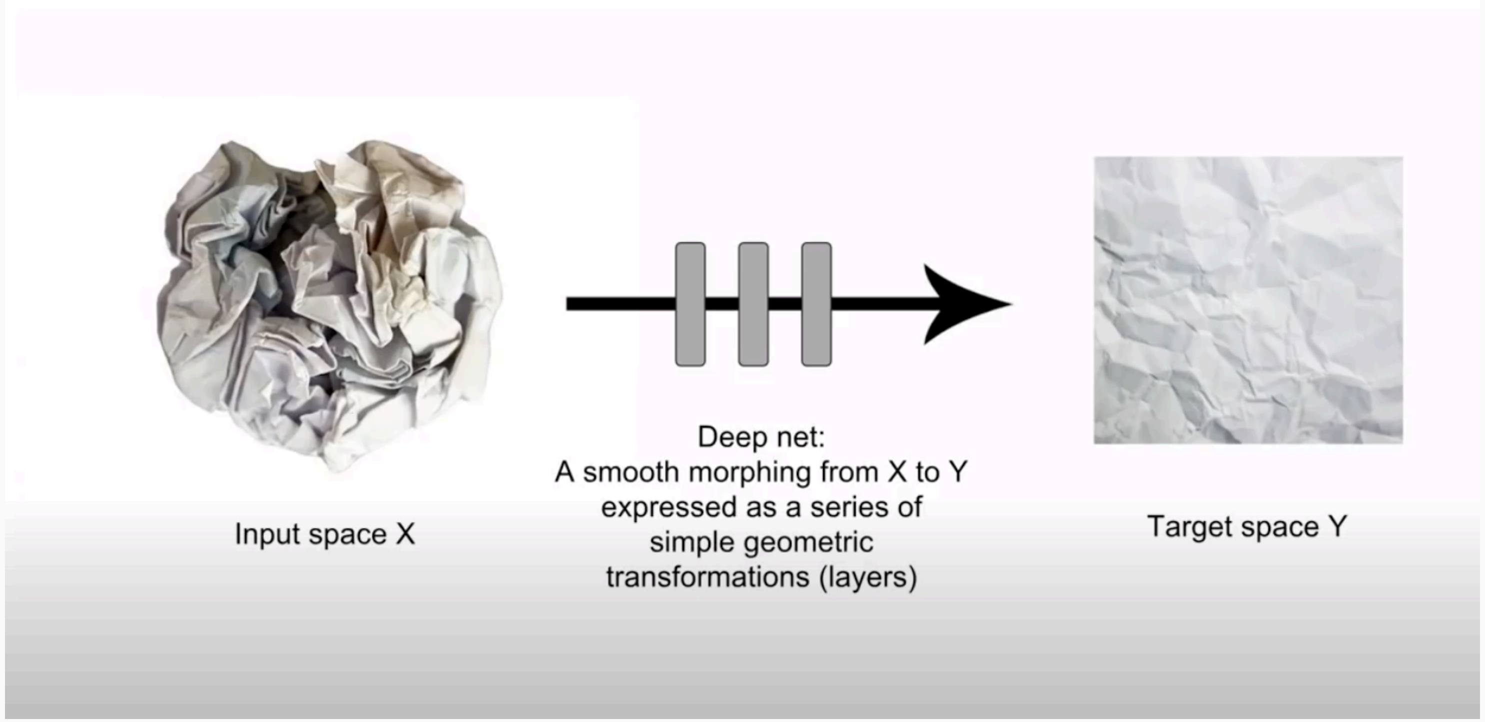
OPTIMIZATION
Performing well on training data



GENERALIZATION
Performing well on unseen data

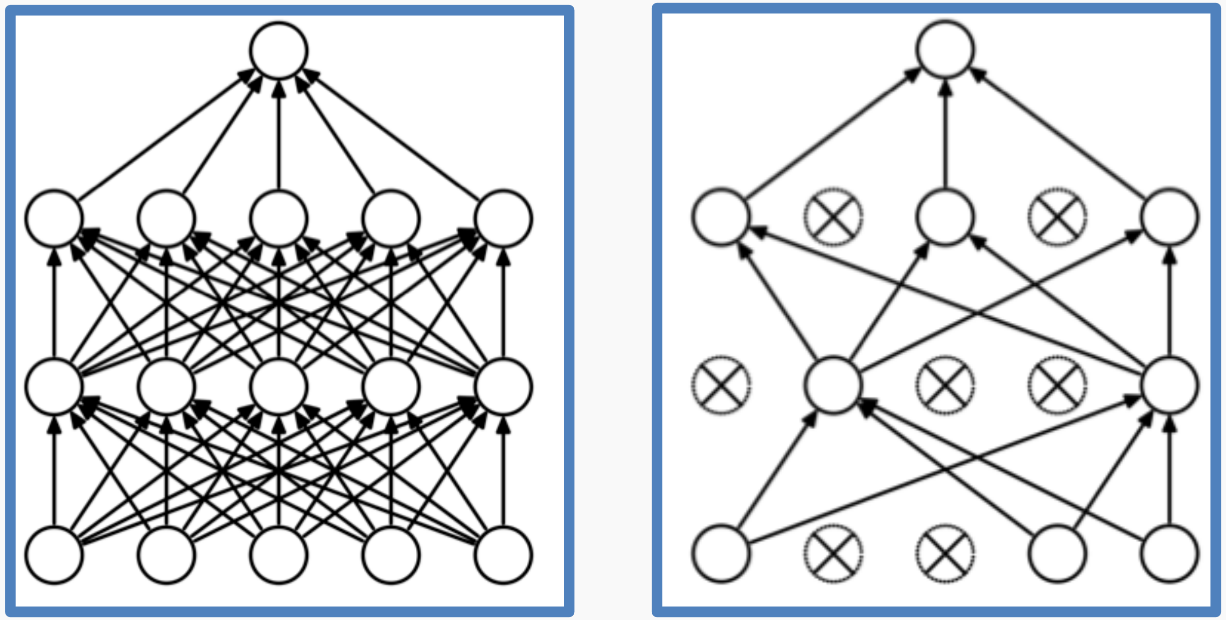
Regularization: Number of layers

We can think of layers as transformations from the input space of the layer to the output space of the layer; a transformation that involves an affine part followed by a non-linear part (activation function). A mapping to a lower dimensional space, as seen in the figure below, from a talk by Francois Chollet, the creator of `keras`.



Regularization : Dropout

Dropout was developed by Geoff Hinton and his students at the University of Toronto. It consists of randomly dropping out (setting to zero) a number of output features of the layer during training. For balance during testing, the layer's output values are scaled down by a factor equal to the dropout rate; no units are dropped out in test time.



In a large network many units can collaborate to respond to the input while the weights can remain relatively small.
This is called co-adaptation.
Dropout prevents overfitting by reducing co-adaptation of neurons. It's like training many random sub-networks.

Regularization : Dropout

Its implementation in `keras` drops a percentage of the input nodes at training time, then scales the values up by the same proportion and does nothing at test time.

```
model.add(layers.Dropout(0.5))
```

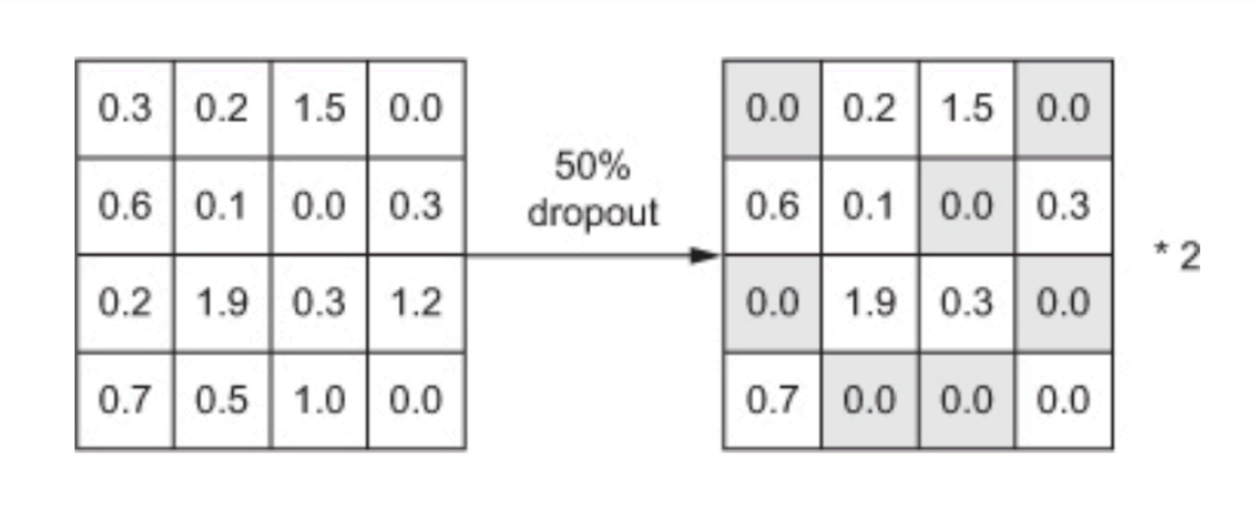
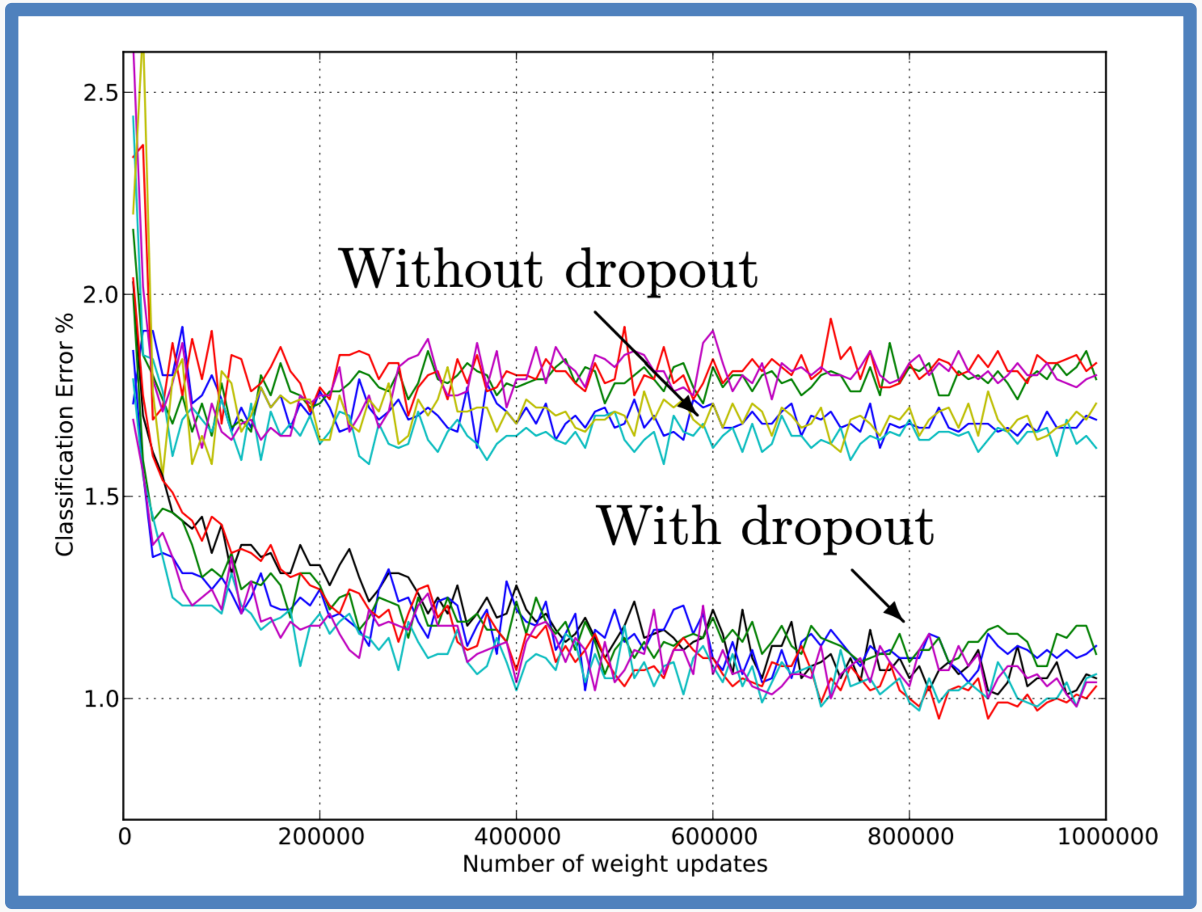


Image: Francois Chollet

Regularization : Dropout

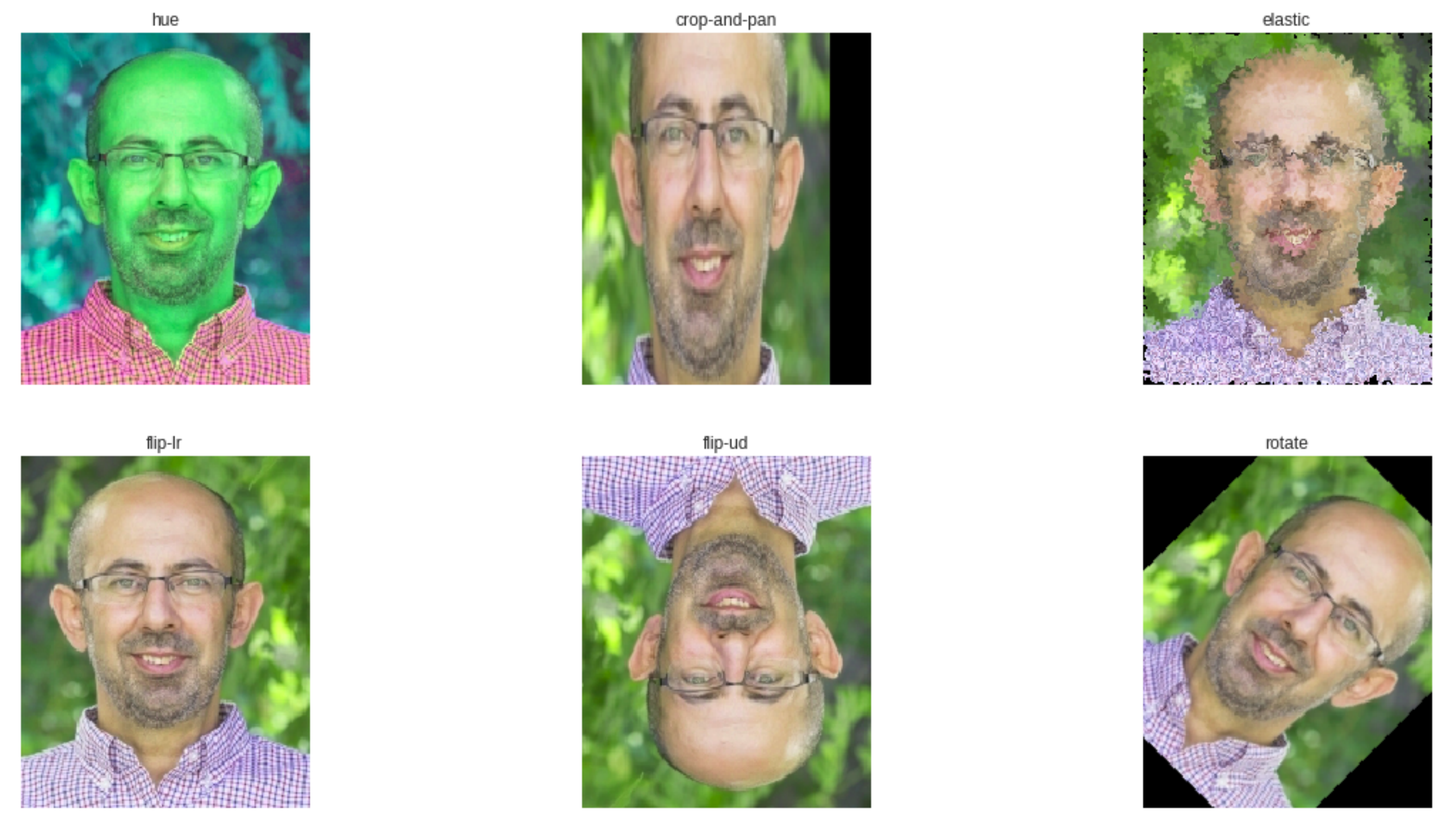
Widely used and highly effective



Test error for different architectures with and without dropout.

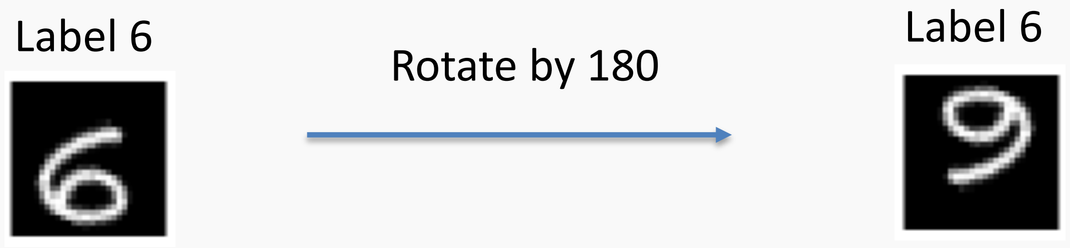
The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Regularization : Data augmentation



Regularization : Data augmentation dos and don'ts

Carefully choose your transformations. Not all transformations are valid.



Data Augmentation does not work for tabular data and not as nicely for time series.

Notebook