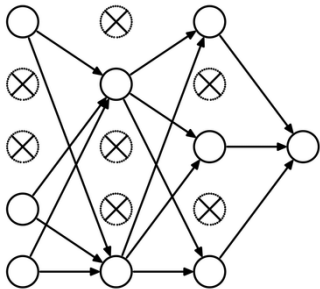


# Deep Learning

## Dropout and Augmentation



### Learning goals

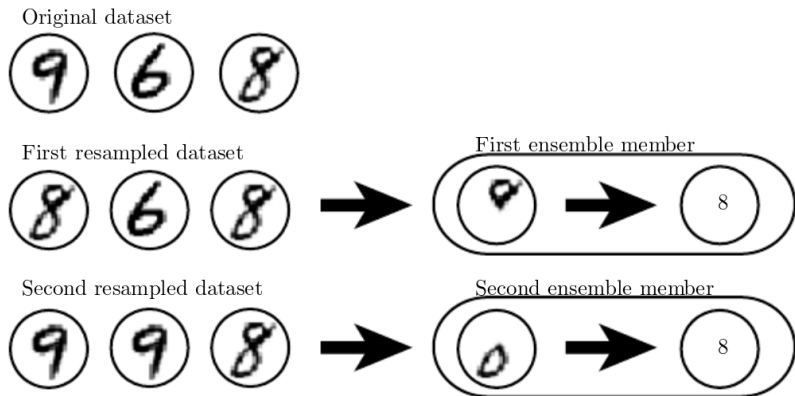
- Recap: Ensemble Methods
- Dropout
- Augmentation

# Ensemble Methods

# RECAP: ENSEMBLE METHODS

- Idea: Train **several models** separately, and **average their prediction** (i.e. perform **model averaging**).
- Intuition: This improves performance on test set, since different models will not make the same errors.
- Ensembles can be constructed in different ways, e.g.:
  - by combining completely different kind of models (using different learning algorithms and loss functions).
  - by **bagging**: train the same model on  $k$  datasets, constructed by sampling  $n$  samples from original dataset.
- Since training a neural network repeatedly on the same dataset results in different solutions (why?) it can even make sense to combine those.

# RECAP: ENSEMBLE METHODS



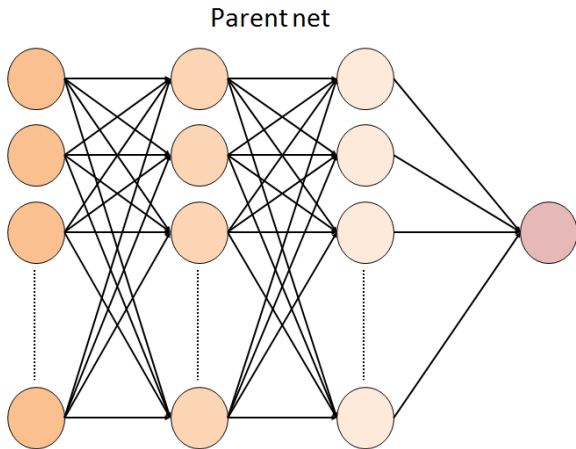
**Figure:** A cartoon description of bagging (Goodfellow et al., 2016)

# Dropout

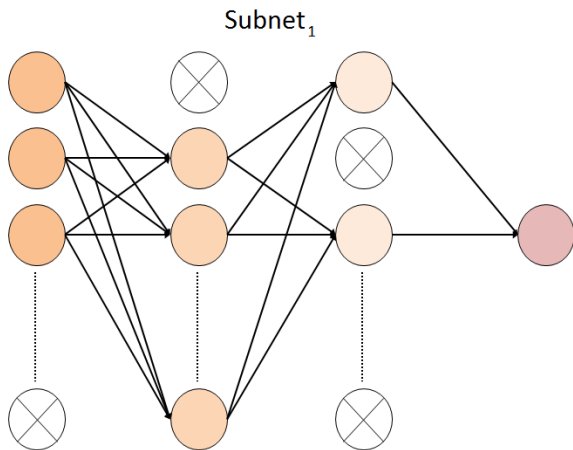
# DROPOUT

- Idea: reduce overfitting in neural networks by preventing complex co-adaptations of neurons.
- Method: during training, random subsets of the neurons are removed from the network (they are "dropped out"). This is done by artificially setting the activations of those neurons to zero.
- Whether a given unit/neuron is dropped out or not is completely independent of the other units.
- If the network has  $N$  (input/hidden) units, applying dropout to these units can result in  $2^N$  possible 'subnetworks'.
- Because these subnetworks are derived from the same 'parent' network, many of the weights are shared.
- Dropout can be seen as a form of "model averaging".

# DROPOUT



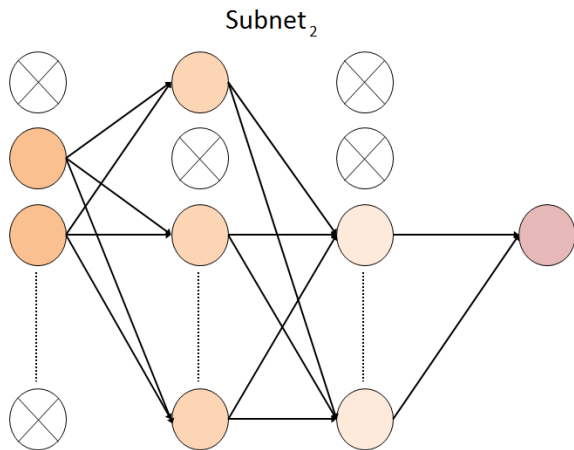
# DROPOUT



In each iteration, for each training example (in the forward pass), a different (random) subset of neurons are dropped out.

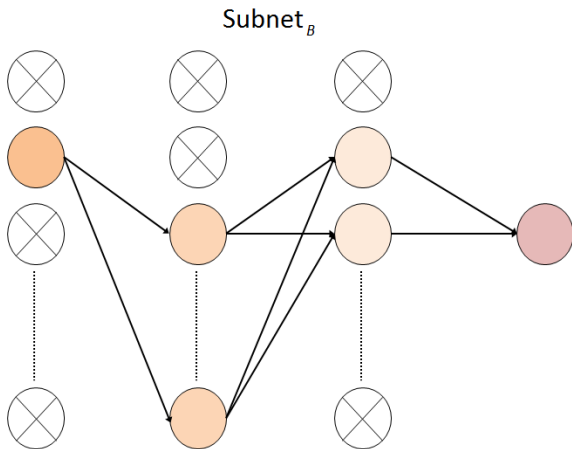


# DROPOUT



In each iteration, for each training example (in the forward pass), a different (random) subset of neurons are dropped out.

# DROPOUT



In each iteration, for each training example (in the forward pass), a different (random) subset of neurons are dropped out.

# DROPOUT: ALGORITHM

- To train with dropout a minibatch-based learning algorithm such as stochastic gradient descent is used.
- For each training case in a minibatch, we randomly sample a binary vector/mask  $\mu$  with one entry for each input or hidden unit in the network. The entries of  $\mu$  are sampled independently from each other.
- The probability  $p$  of sampling a mask value of 0 (dropout) for one unit is a hyperparameter known as the 'dropout rate'.
- A typical value for the dropout rate is 0.2 for input units and 0.5 for hidden units.
- Each unit in the network is multiplied by the corresponding mask value resulting in a *subnet* $_{\mu}$ .
- Forward propagation, backpropagation, and the learning update are run as usual.

# DROPOUT: ALGORITHM

---

**Algorithm 1** Training a (parent) neural network with dropout rate  $p$ 

---

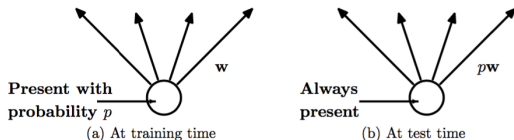
```
1: Define parent network and initialize weights
2: for each minibatch: do
3:   for each training sample: do
4:     Draw mask  $\mu$  using  $p$ 
5:     Compute forward pass for  $subnet_{\mu}$ 
6:   end for
7:   Update the weights of the (parent) network by performing a gradient descent step
    with weight decay
8: end for
```

---

- The derivatives wrt. each parameter are averaged over the training cases in each mini-batch. Any training case which does not use a parameter contributes a gradient of zero for that parameter.

# DROPOUT: WEIGHT SCALING

- The weights of the network will be larger than normal because of dropout. Therefore, to obtain a prediction at test time the weights must be first scaled by the chosen dropout rate.
- This means that if a unit (neuron) is retrained with probability  $p$  during training, the weight at test time of that unit is multiplied by  $p$ .



**Figure:** Training vs. Testing (Srivastava et. al., 2014)

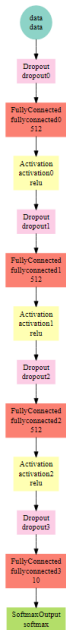
- Weight scaling ensures that the expected total input to a neuron/unit at test time is roughly the same as the expected total input to that unit at train time, even though many of the units at train time were missing on average

# DROPOUT: WEIGHT SCALING

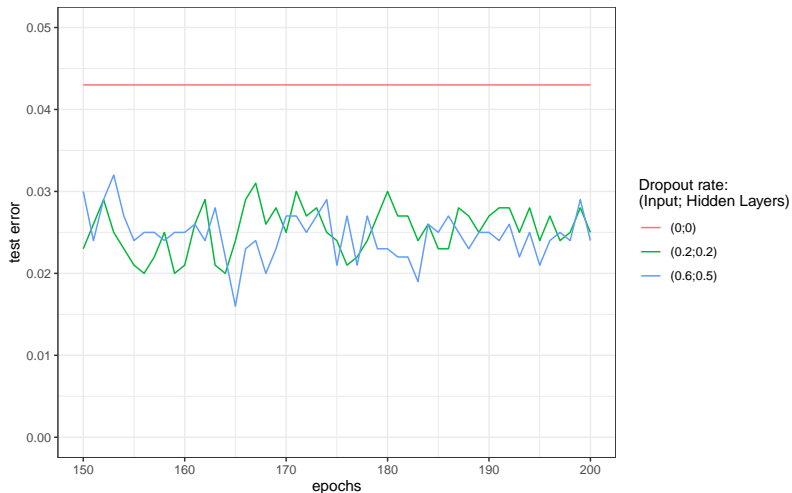
- Rescaling of the weights can also be performed at training time instead, after each weight update at the end of the mini-batch. This is sometimes called 'inverse dropout'. Keras and PyTorch deep learning libraries implement dropout in this way.

# DROPOUT: EXAMPLE

- To demonstrate how dropout can easily improve generalization we compute neural networks with the structure showed on the right.
- Each neural network we fit has different dropout probabilities, a tuple where one probability is for the input layer and one is for the hidden layers. We consider the tuples  $(0; 0)$ ,  $(0.2; 0.2)$  and  $(0.6; 0.5)$ .



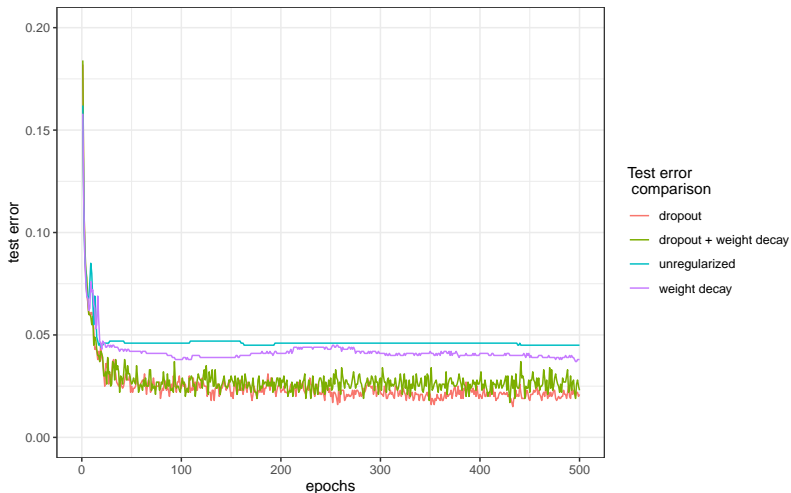
# DROPOUT: EXAMPLE



Dropout rate of 0 (no dropouts) leads to higher test error than dropping some units out.



# DROPOUT, WEIGHT DECAY OR BOTH?



Here, dropout leads to a smaller test error than using no regularization or solely weight decay.

# Dataset Augmentation

# DATASET AUGMENTATION

- Problem: low generalization because high ratio of

$$\frac{\text{complexity of the model}}{\text{\#train data}}$$

- Idea: artificially increase the train data.
  - Limited data supply → create “fake data”!
- Increase variation in inputs **without** changing the labels.
- Application:
  - Image and Object recognition (rotation, scaling, pixel translation, flipping, noise injection, vignetting, color casting, lens distortion, injection of random negatives)
  - Speech recognition (speed augmentation, vocal tract perturbation)

# DATASET AUGMENTATION



(a) Original



(b) Color



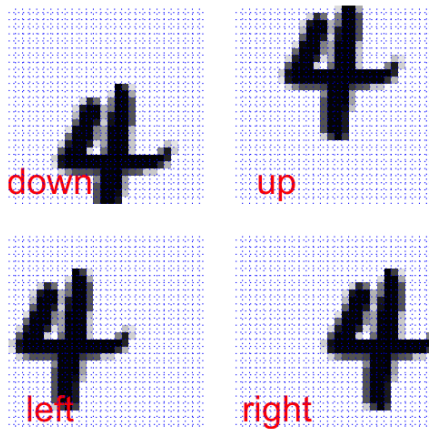
(c) Rotate



(d) Horizontal Stretch

**Figure:** Example for data set augmentation (Wu et al., 2015)





# DATASET AUGMENTATION



**Figure:** Example for data set augmentation (Wu et al., 2015)

⇒ careful when rotating digits (6 will become 9 and vice versa)!

# REFERENCES

-  Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
-  Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors.
-  Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56), 1929–1958.  
<http://jmlr.org/papers/v15/srivastava14a.html>
-  Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep Image: Scaling up Image Recognition.