



Some Best Practices for Machine Learning

(In Chemistry/Materials Science)

Some good resources

- Intro textbooks all provide a reasonable framework to understand the basics of obviously wrong/good things to do

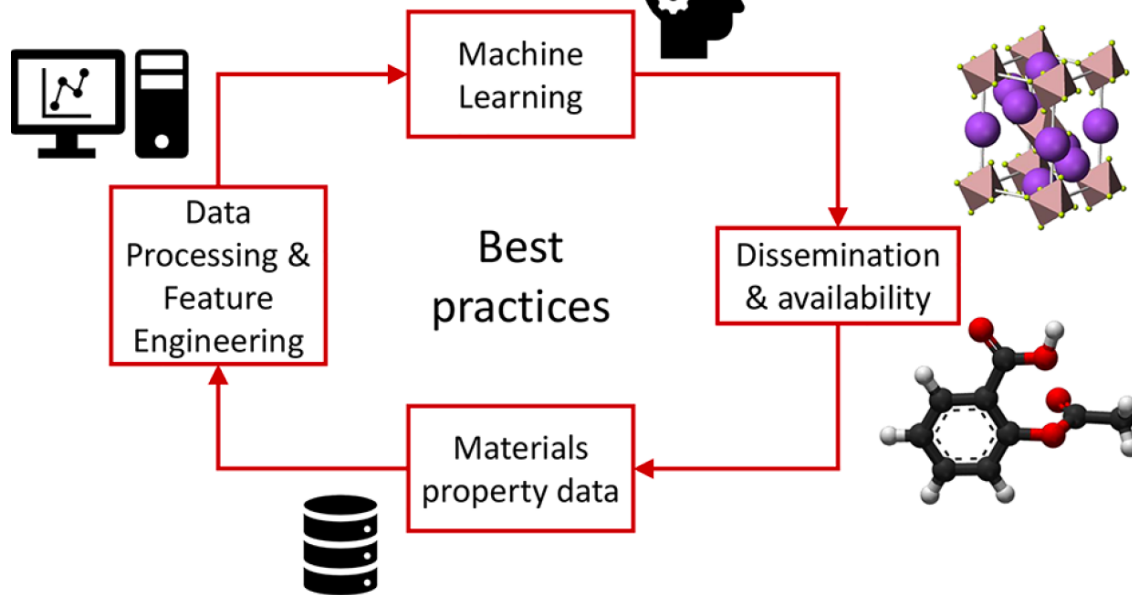
Some more targeted discussion:

Machine Learning for Materials Scientists: An Introductory Guide toward Best Practices

Anthony Yu-Tung Wang, Ryan J. Murdock, Steven K. Kauwe, Anton O. Oliynyk, Aleksander Gurlo, Jakoah Brgoch, Kristin A. Persson, and Taylor D. Sparks*

✓ Cite This: *Chem. Mater.* 2020, 32, 4954–4965

Read Online



Best practices in machine learning for chemistry

Nongnuch Artrith, Keith T. Butler, François-Xavier Coudert, Seungwu Han, Olexandr Isayev, Anubhav Jain and Aron Walsh

- Best practices need not/cannot be universal practices
- There are a lot of people doing wonky things out there
- Some advice may not be practicable

What is FAIR?

Artificial intelligence faces reproducibility crisis

Matthew Hutson

[+ See all authors and affiliations](#)

Science 16 Feb 2018:
Vol. 359, Issue 6377, pp. 725-726
DOI: 10.1126/science.359.6377.725

Article **Figures & Data** **Info & Metrics** **eLetters**  **PDF**

Summary

The booming field of artificial intelligence (AI) is grappling with a replication crisis, much like the ones that have afflicted psychology, medicine, and other fields over the past decade. Just because algorithms are based on code doesn't mean experiments are easily replicated. Far from it. Unpublished codes and a sensitivity to training conditions have made it difficult for AI researchers to reproduce many key results. That is leading to a new conscientiousness about research methods and publication protocols. Last week, at a meeting of the Association for the Advancement of Artificial Intelligence in New Orleans, Louisiana, reproducibility was on the agenda, with some teams diagnosing the problem—and one laying out tools to mitigate it.


[Open Access](#) | [Published: 15 March 2016](#)

The FAIR Guiding Principles for scientific data management and stewardship

Mark D. Wilkinson, Michel Dumontier, [...]Barend Mons 

[Scientific Data](#) **3**, Article number: 160018 (2016) | [Cite this article](#)

327k Accesses | **2821** Citations | **1881** Altmetric | [Metrics](#)

 An [Addendum](#) to this article was published on 19 March 2019

Abstract

There is an urgent need to improve the infrastructure supporting the reuse of scholarly data. A diverse set of stakeholders—representing academia, industry, funding agencies, and scholarly publishers—have come together to design and jointly endorse a concise and measureable set of principles that we refer to as the FAIR Data Principles. The intent is that these may act as a guideline for those wishing to enhance the reusability of their data holdings. Distinct from peer initiatives that focus on the human scholar, the FAIR Principles put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals. This Comment is the first formal publication of the FAIR Principles, and includes the rationale behind them, and some exemplar implementations in the community.

FAIR data principles are aimed at helping reduce this reproducibility crisis by abiding by a set of concrete standards across disciplines.

Possible venues for data deposition

If there comes a time to publish, you should consider resources that make things more FAIR



NIST Materials Data Repository



DLHub



 DataSpace



Publishing your data/code is the way

- Existing tools make this relatively painless

[DataSpace](#) [About](#) [Contact](#) [Help](#)

[Home](#) / [Chemical and Biological Engineering](#) / [Research Data Sets](#)

Please use this identifier to cite or link to this item: <http://arks.princeton.edu/ark:/88435/dsp01765374506>

Title: **Data for Coarse-grained Intrinsically Disordered Proteins** **#regret**

Contributors: Webb, Michael
Patel, Roshan
Borca, Carlos

Keywords: [Machine Learning](#)
[Protein](#)
[Molecular Simulation](#)
[Molecular Dynamics](#)

Issue Date: 6-May-2022

Publisher: Princeton University

Abstract: This distribution compiles numerous physical properties for 2,585 intrinsically disordered proteins (IDPs) obtained by coarse-grained molecular dynamics simulation. This combination comprises "Dataset A" as reported in "Featurization strategies for polymer sequence or composition design by machine learning" by Roshan A. Patel, Carlos H. Borca, and Michael A. Webb (DOI: 10.1039/D1ME00160D). The specific IDP sequences are sourced from version 9.0 of the DisProt database. The simulations were performed using the LAMMPS molecular dynamics engine. The interactions used for simulation are obtained from R. M. Regy , J. Thompson , Y. C. Kim and J. Mittal , Improved coarse-grained model for studying sequence dependent phase separation of disordered proteins, Protein Sci., 2021, 1371 — 1379.

URI: <http://arks.princeton.edu/ark:/88435/dsp01765374506>
<https://doi.org/10.34770/chzn-mj42>

Referenced By: <https://doi.org/10.1039/D1ME00160D>

Appears in Collections: [Research Data Sets](#)

Files in This Item:

File	Description	Size	Format	
README		2.67 kB	Text	View/Download
dataset_a_sequences.txt		654.7 kB	Text	View/Download
dataset_a_encodings.csv		113.97 kB	CSV	View/Download
dataset_a_labels.csv		89.81 kB	CSV	View/Download

- It is increasingly viewed as good practice – referees should recognize when you have data of interest and request you do this; journals may even require it!

personal examples:

The editorial office at Science Advances has completed a technical check on your revised manuscript. Your submission is being returned to you and must be corrected before moving forward. The reasons are listed below:

- Please replace the currently uploaded combined PDF with a copy that includes clearly marked/highlighted revisions (NOT track changes, i.e., no crossed-out text). You may do this by either highlighting the text, or simply changing the color of the font. This is essential to the reviewers and editors during the review process. Please note, however, that the individual manuscript and supplementary files should not have these highlights.
- We note that your Data and Materials Availability statement contains a link to data deposited exclusively on [GitHub](#). Upon publication, data and code must be available in the paper or archived in a permanent, independent, preferably non-profit repository such as Dryad, Dataverse, or Zenodo with an access code provided in the Data and Materials Availability statement. A [GitHub](#) link may be included but only as an additional resource. For additional information, please see our editorial policies website here: <https://www.sciencemag.org/authors/science-journals-editorial-policies#data-and-code-deposition>.
- Please amend your Data and Materials Availability section by revising the following sentence: "Additional (raw) data not distributed may be requested from the corresponding authors." Per our updated open access policy, statements such as "available upon request" are not permitted. If any additional data exists that is not contained within the manuscript or supplementary materials, it must be archived in a permanent repository such as Dryad, Dataverse, or Zenodo. If utilized, please add a direct link to the archived data in your Data and Materials Availability section.

- (4) "Finally, I think given there is a strong technical component here, I'd encourage the authors to deposit working code that demonstrates some of the analysis used here in a way that a technical user could take that code and adapt it to their own system question. This could just be a jupyter notebook that reads in simulation data and calculates B2 or the Empirical EOS curves in Fig 1."

Author Reply: *This is a good idea. We have deposited our data to a public repository. We have also uploaded demonstrative python scripts/Jupyter notebooks to Github. These additions have been noted.*

A critical look at data

All problems need to start by carefully considering the source of data



Knowing what the data is/where it was created may dictate reproducibility

Are all data sources listed and publicly available?
If the data is yours, how will it be communicated?



Most databases change over time.

If using an external database, is an access date or version number provided?



All data likely has bias, and that is not a problem, but the bias must be clearly identified and stated.

Are any potential biases in the source dataset reported and/or mitigated?
Even if not reported, can they be ascertained?

A critical look at data

Sanitization of datasets is common but dangerous

Common pre-processing steps:

- removing duplicates
- addressing missing values
- removing unphysical values
- identifying outliers*
- applying feature scaling techniques

Questions to consider:

- Are the data cleansing steps clearly and fully described? (textual descriptions need to be complete! if overly cumbersome, then is there a code to demonstrate the pipeline?→reproducibility!!!!)
- Is there an evaluation of the impact of the data that was removed? Comparative analysis is often warranted.
- Are data combined from disparate sources? Are there steps to mitigate potential issues with their merger?

A critical look at data

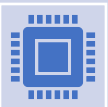
Take time to try to understand your data before fitting it!



*Remember that your ML model is only as good as your data. Take some time to analysis the statistical properties of your data before pulling out the ML models. **You may learn a lot!***



Are there gaps in your data? Missing data points? How do you fill these in? These are important questions that can strongly bias results.



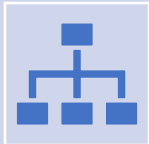
If there is intrinsic noise in your data set, then if your ML models shows $r^2=1$ and a perfect fit, something is terribly wrong.



Similarly, if you attempt to fit a model, and you obtain universal predictions (probably the mean), something has gone awry

A critical look at data

Take time to try to understand your data before fitting it!



What is the structure of your data?



Does your dataset form clusters based on chemical formula, test condition, or other criteria?



Are some clusters over/under-represented?

We will later explore unsupervised learning techniques (PCA, t-SNE, UMAP) to visualize your data structure.

Considering featurization

We spent a fair bit of time talking about this, but just to recap:

- *The choice of representation can be as important as the choice of machine learning method – probably more important (don't set yourself up for failure!)*
- *Examine your data before doing anything to it.*
 - *Is it distributed in any special way?*
 - *Does it have natural bounds?*
 - *Are there positive and negative values?*
 - *Do particular values exhibit special meaning?*
- *It is fair to evaluate different representations when constructing models. We can think of the approach to representation like a hyperparameter.*
- *In many chemical/materials systems, the data may possess certain equivariances or invariances based on physical characteristics of the system.*
 - *Does the representation method account for these?*
 - *Are data augmentation methods viable?*
- *Making things complex does not mean making them better. You should compare your approach to established methods and prove that they are better.*



If using data transformation, it is best to only apply/train the transform to your **training data**. Don't forget to inverse transform too!

Considering featurization

We spent a fair bit of time talking about this, but just to recap:

- *Start by sourcing ideas/representation schemes from the literature. People have spent a lot of time already thinking about this.*
- *If there is a viable one-hot encoding for your system... consider it as a baseline.*

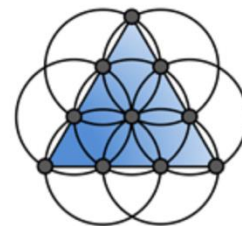
Some useful resources:



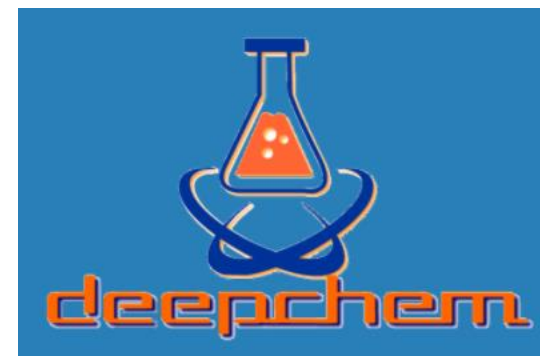
Mordred

03334432 023333321
1411222243 33111222244
2301222225 3311122222333
23322211344344443 22 0222222343
01213533334444443245
53333444445556664
5212344445556665
43 144445556672
34234455556662
0345555666530
01110

DScribe



matminer



Considering featurization

Featurizers

DeepChem contains an extensive collection of featurizers. If you haven't run into this terminology before, a "featurizer" is chunk of code which transforms raw input data into a processed form suitable for machine learning. Machine learning methods often need data to be pre-chewed for them to process. Think of this like a mama penguin chewing up food so the baby penguin can digest it easily.

```
>>> import deepchem as dc
>>> from rdkit import Chem
>>> smiles = ['C1=CC=CC=C1']
>>> # Example 1: (size = 2048, radius = 4)
>>> featurizer = dc.feat.CircularFingerprint(size=2048, radius=4)
>>> features = featurizer.featurize(smiles)
>>> type(features[0])
<class 'numpy.ndarray'>
>>> features[0].shape
(2048,)
```

- Molecule Featurizers
 - Graph Convolution Featurizers
 - ConvMolFeaturizer
 - WeaveFeaturizer
 - MolGanFeaturizer
 - MolGraphConvFeaturizer
 - PagtnMolGraphFeaturizer
 - DMPNNFeaturizer
 - GroverFeaturizer
 - RDKitConformerFeaturizer
 - MXMNetFeaturizer
 - Utilities
 - MACCSKeysFingerprint
 - MATFeaturizer
 - CircularFingerprint
 - PubChemFingerprint
 - Mol2VecFingerprint
 - RDKitDescriptors
 - MordredDescriptors
 - CoulombMatrix
 - CoulombMatrixEig
 - AtomCoordinates
 - BPSymmetryFunctionInput
 - SmilesToSeq
 - SmilesToImage
 - OneHotFeaturizer
 - SparseMatrixOneHotFeaturizer
 - RawFeaturizer
 - SNAPFeaturizer
- Molecular Complex Featurizers
 - RdkitGridFeaturizer
 - AtomicConvFeaturizer
- Inorganic Crystal Featurizers
 - MaterialCompositionFeaturizer
 - ElementPropertyFingerprint
 - ElemNetFeaturizer
 - MaterialStructureFeaturizer
 - SineCoulombMatrix
 - CGCNNFeaturizer
 - LCNNFeaturizer
- Molecule Tokenizers
 - SmilesTokenizer
 - BasicSmilesTokenizer
 - HuggingFaceFeaturizer
 - GroverAtomVocabTokenizer
 - GroverBondVocabTokenizer
- Vocabulary Builders
- Sequence Featurizers
 - PFMFeaturizer
- Other Featurizers
 - BertFeaturizer
 - RobertaFeaturizer
 - RxnFeaturizer
 - BindingPocketFeaturizer
 - UserDefinedFeaturizer
 - DummyFeaturizer
- Base Featurizers (for develop)
 - Featurizer
 - MolecularFeaturizer
 - MaterialCompositionFeaturizer
 - MaterialStructureFeaturizer
 - ComplexFeaturizer
 - VocabularyBuilder
 - HuggingFaceVocabularyBuilder

Selecting a machine learning model

For each problem, spend some time considering the algorithms themselves and their hyperparameters

- ***Is a software implementation of the model provided such that it can be trained and tested with original data or new data?***
 - GitHub is a great resource here.
 - One should strive to report every aspect of your model required for reproduction (hyperparameters, neural network weights, etc).
- ***Are baseline comparisons to simple/trivial models (e.g. 1-nearest neighbor, random forest, linear regression) provided?***
- ***Are baseline comparisons to current state-of-the-art provided? Relevant?***

Selecting a machine learning model

Hyperparameter optimization is often a necessary evil; at the very least, it helps to promote model robustness!

Points for consideration:

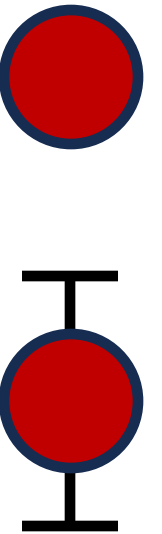
Just because your fancy model does terribly the first time, doesn't mean it won't eventually work – perform hyperparameter optimization!

Before going ham on the hyperparameter optimization, spend some time tinkering with the parameters to gain intuition for reasonable ranges. Use a subset of your data if necessary.

Consider also the relationship between your data and the model complexity. This goes back to examining your data! Can you justify a model with so many parameters with the amount and structure of data?

*Grid search, random sampling, and Bayesian techniques all work. Use existing packages (e.g. **Hyperopt**, **Optuna**) to automate this process.*

Be sure that you check the performance on a held-out test set! It is easy to induce bias by overfitting during hyperparameter optimization. Nested k-fold may be appropriate.



Selecting a machine learning model

Hyperparameter optimization is often a necessary evil; at the very least, it helps to promote model robustness!

For model evaluation:

- *Don't rely on a single metric, if possible. Keep in mind the biases of any given metric. **What do they tell you? What are they sensitive to?***
- *Don't just trust the numbers. Look again at the data.*
- *It is essentially free to compute a variety of different error metrics. **Why not do so if sensible?***

Reporting/Evaluating a machine learning model

Checklist for reporting and evaluating machine learning models	
1. Data sources	
1a. Are all data sources listed and publicly available?	
1b. If using an external database, is an access date or version number provided?	
1c. Are any potential biases in the source dataset reported and/or mitigated?	
2. Data cleaning	
2a. Are the data cleaning steps clearly and fully described, either in text or as a code pipeline?	
2b. Is an evaluation of the amount of removed source data presented?	
2c. Are instances of combining data from multiple sources clearly identified, and potential issues mitigated?	
3. Data representations	
3a. Are methods for representing data as features or descriptors clearly articulated, ideally with software implementations?	
3b. Are comparisons against standard feature sets provided?	
4. Model choice	
4a. Is a software implementation of the model provided such that it can be trained and tested with new data?	
4b. Are baseline comparisons to simple/trivial models (for example, 1-nearest neighbour, random forest, most frequent class) provided?	
4c. Are baseline comparisons to current state-of-the-art provided?	
5. Model training and validation	
5a. Does the model clearly split data into different sets for training (model selection), validation (hyperparameter optimization), and testing (final evaluation)?	
5b. Is the method of data splitting (for example, random, cluster- or time-based splitting, forward cross-validation) clearly stated? Does it mimic anticipated real-world application?	
5c. Does the data splitting procedure avoid data leakage (for example, is the same composition present in training and test sets)?	
6. Code and reproducibility	
6a. Is the code or workflow available in a public repository?	
6b. Are scripts to reproduce the findings in the paper provided?	

Overfitting



M

Why does overfitting happen in machine learning models?



Overfitting occurs in machine learning models due to the model's ability to learn not only the underlying patterns in the data but also the noise or random fluctuations present in the training data. This happens because machine learning models, especially complex ones, have the capacity to fit the training data extremely closely, sometimes to the point of capturing data points that don't represent true underlying patterns. Here are some of the key reasons why overfitting happens:

1. **Model Complexity:** Complex machine learning models, such as deep neural networks with many layers and parameters, have a high capacity to fit the training data. While this allows them to capture intricate patterns, it also makes them more prone to overfitting.
2. **Limited Data:** When the training dataset is small or not representative of the entire population, the model may not have enough information to generalize effectively. In such cases, the model may resort to memorizing the training data, leading to overfitting.
3. **Noisy Data:** If the training data contains noise, errors, or outliers, the model may mistakenly capture and fit to these noisy data points rather than focusing on the true underlying patterns.
4. **Irrelevant Features:** Including irrelevant features in the input data can confuse the model and lead to overfitting. The model may attempt to learn patterns from irrelevant features, which can hurt its ability to generalize.
5. **Insufficient Regularization:** Regularization techniques, such as L1 or L2 regularization, help control overfitting by penalizing overly complex model parameters. If these techniques are not properly applied or tuned, the model may not be regularized effectively, allowing it to overfit.
6. **Lack of Validation:** In some cases, a model's performance is only evaluated on the training data, and there is no separate validation set to assess its generalization capabilities. Without validation, it's challenging to identify overfitting.

- *OK, don't use models with more parameters than warranted by the amount of data. (fewer layers, fewer neurons per layer, fewer trees, etc.)*
- *Having limited data is not permission to overfit. Having insufficient data should yield a rightfully bad model not a misleadingly good one.*
- *If you are training your model on data not representative of the entire population, then your model is not overfitted, it is just irrelevant (to that domain).*
- *Models can only learn from the data provided. Noise can be addressed in different ways and uncertainty assessed.*
If your data is erroneous, that's a data problem, not a model problem.
- *Don't include irrelevant features. They are irrelevant.*
- *Use sufficient regularization.*
- *Validate.*

Overfitting

In addition to aforementioned considerations, there are several steps one can take to mitigate overfitting

- **Cross-Validation** – e.g., k -fold cross-validation. Let one of the groups be the testing set and the others groups collected together as the training set. Repeat this process until each individual group has been used as testing.
- **Data augmentation** – A larger dataset will reduce overfitting. If more data is hard to come by, we can apply data augmentation to artificially increase the size of the dataset. This is particularly useful for image recognition in which one can add training data in which the original representation has been e.g. flipped, rotated, shifted, or scaled.
- **Feature Selection** – If there are many features used as input, we should only select the most important features so that the model can reduce the scope of parameters it must fit. Calculate F statistics – uses variance between features and variance within each feature.

Overfitting

In addition to aforementioned considerations, there are several steps one can take to mitigate overfitting

- **Consider regularization**

- L1 penalizes sum of the absolute value of model weights. Usually aggressively eliminates extraneous features. Robust to outliers. (**LASSO**)
- L2 penalizes sum of square value of weights. Not as robust to outliers. (**Ridge**)

- **Simplify your model**

Consider penalties for model complexity. If it's a neural network, remove layers or reduce their size. You wouldn't try to fit a 50th order polynomial to 6 data points!

- **Apply dropout**

A form of regularization applied to neural network layers in which we ignore a subset of neurons with a set probability. This can reduce interdependent learning among units during training, which can lead to overfitting.

- **Stop early**

Early stopping is an actual thing. Monitor validation or test loss by comparison to training loss. Stop your parameterization when validation/test loss is near the minimum.

Some advice on neural networks

A rough priority ranking for tuning neural network performance according to Andrew Ng



- Cofounder and head of Google Brain
- Chief Scientist at Baidu
- Founder of Coursera
- Founder of DeepLearning.ai
- Professor at Stanford
- Knows what the hell he's talking about

decreasing importance

Learning rate
Momentum beta
Mini-batch size
of hidden units
of layers
Learning rate decay
Regularization lambda
Activation functions
Adam's beta1 & beta2

Additional empirical thoughts:

- **batch normalization**, feature-scaling, cross-validation are pretty much always good things to do
- funnel structures are good starting points
- ReLU and Elu are good starting points
- ADAM (+nesterov) is usually best optimizer
- learning rate is important

Dropout Regularization

Idea: During training, set a probability, p , that some number of nodes within layers are randomly ignored or “dropped out.”

- Dropout has the effect of making the training process noisy.
- Dropout leads to a sparsification of the hidden unit weights, similar to L2 regularization.
- Dropout is not used when making predictions at test time.

Inverted Dropout (Most Common): Weights of the network will be larger than normal because of dropout, so scale the weights by the chosen dropout rate, p .

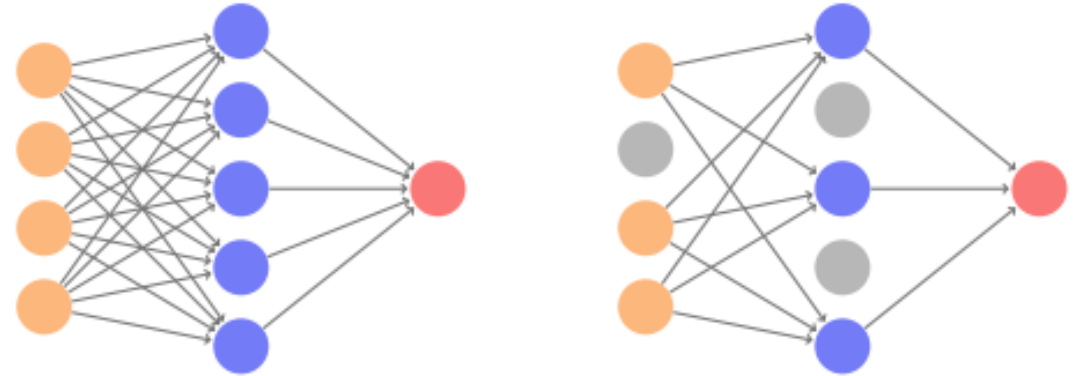


FIGURE 10.19. *Dropout Learning. Left: a fully connected network. Right: network with dropout in the input and hidden layer. The nodes in grey are selected at random, and ignored in an instance of training.*

Adding a dropout layer in Keras:

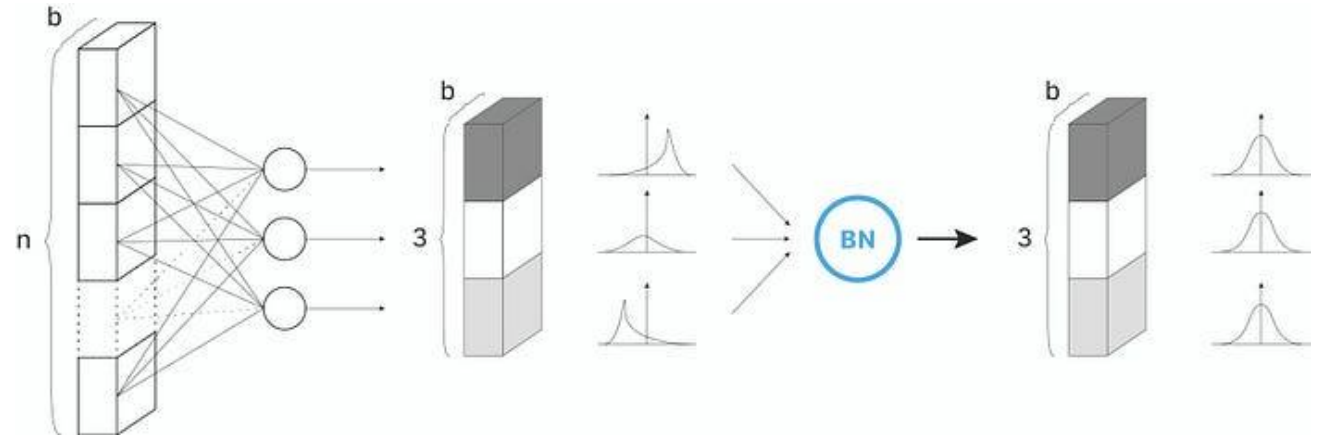
```
model.add(Dropout(0.2))
```

Batch Normalization

Theoretical Problem: Each mini-batch the neural network sees may have a different looking distribution, *causing the learning algorithm to chase a moving target (internal covariate shift).*

Idea: Standardize the inputs to a layer for each mini-batch.

- Critical for training very deep networks.
- Faster and smoother convergence.
- Allows one to use a larger learning rate.
- Has a minor regularizing effect on the network!



```
layers.BatchNormization()
```

Ioffe & Szegedy "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167

Image: <https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338>

Batch Normalization

During Training: At each hidden layer, Batch Normalization transforms the signal using

$$(1) \mu = \frac{1}{n} \sum_i Z^{(i)} \quad (2) \sigma^2 = \frac{1}{n} \sum_i (Z^{(i)} - \mu)^2$$

$$(3) Z_{norm}^{(i)} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 - \epsilon}} \quad (4) \check{Z} = \gamma * Z_{norm}^{(i)} + \beta$$

After calculating the mean and standard deviation of the batch, BatchNorm applies a linear transformation with **learnable parameters** gamma and beta

Always apply batch norm **after** activation for best results!

During Evaluation: Compute the population level mean and standard deviation and use that across all data predictions.

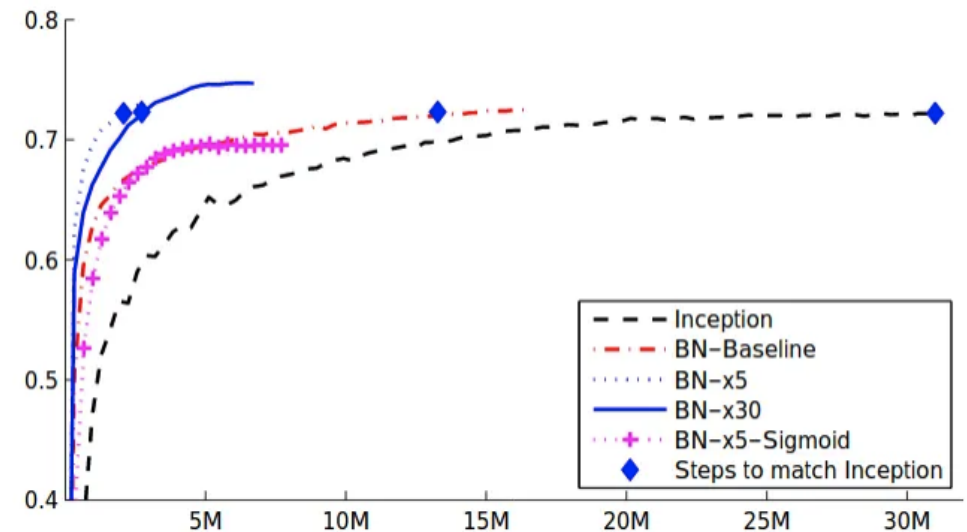


Figure 5: Batch normalization impact on training (ImageNet) | "Inception": original network [3]; "BX-Baseline": same as Inception with BN, same learning rate (LR); "BX-x5": same as Inception with BN, LR x5; "BX-x30": same as Inception with BN, LR x30; "BN-x5-Sigmoid": same as Inception with BN, LR x5 and sigmoid instead of ReLU | Source: [1]

Ioffe & Szegedy "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167

Image: <https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338>