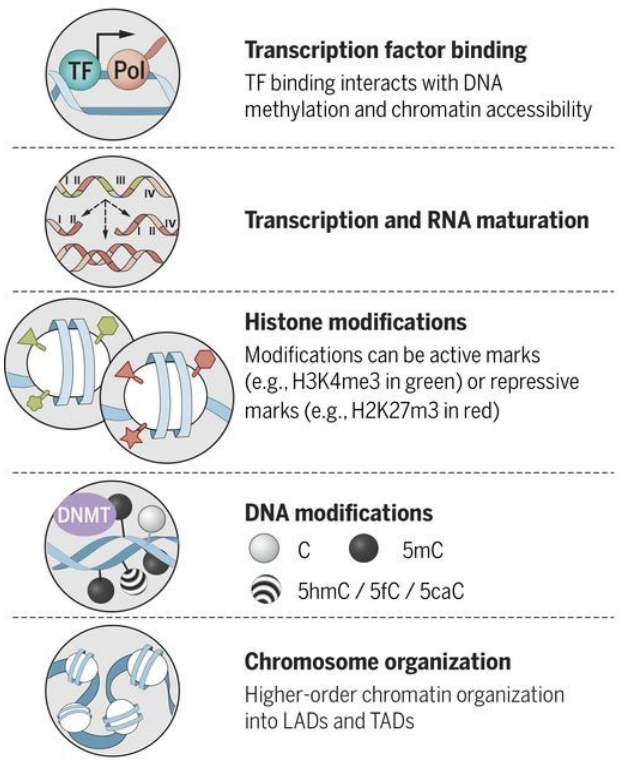
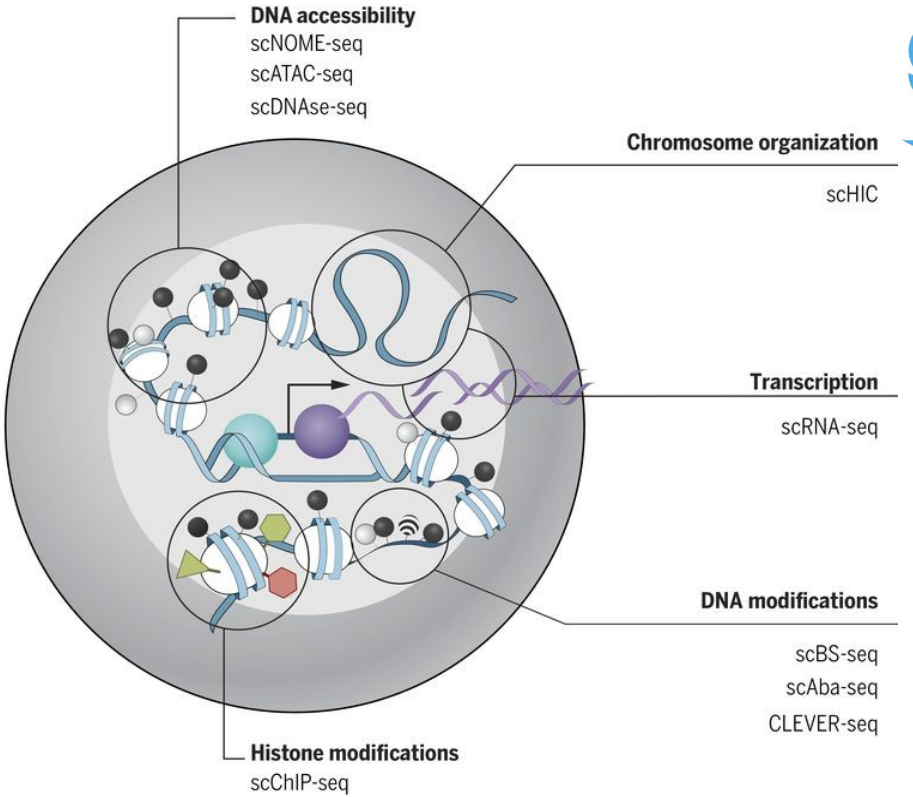
Artificial Neural Networks for MultiOmics Integration

Physalia course, online via zoom

Nikolay Oskolkov, MRG Group Leader, LIOS, Riga, Latvia





@NikolayOskolkov

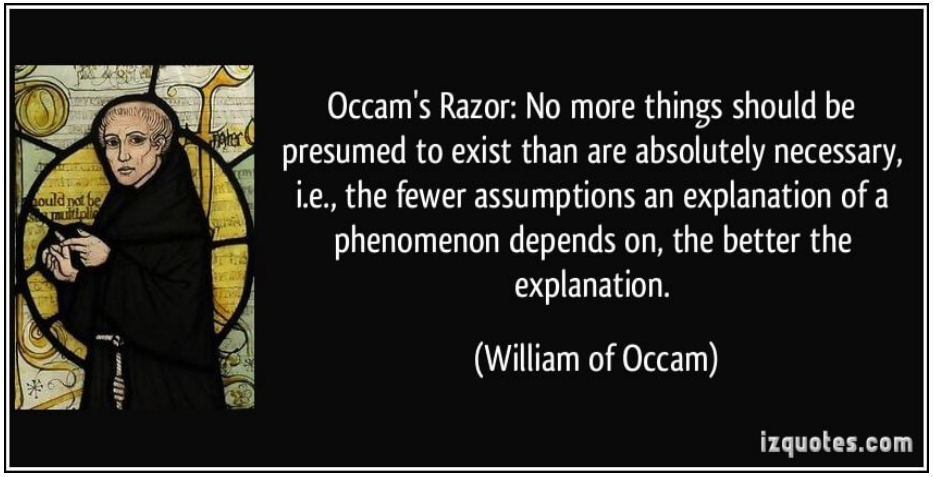
@oskolkov.bsky.social

Personal homepage:
https://nikolay-oskolkov.com

Image adapted from Kelsey et al., Science 2017

Challenges of Deep Learning in Life Sciences

- Difficult to apply to real Life Science projects (NGS: tabular data)
- Lack of data in Life Sciences (exceptions: single cell, microscopy)
- Simpler (than Deep Learning) methods often perform better

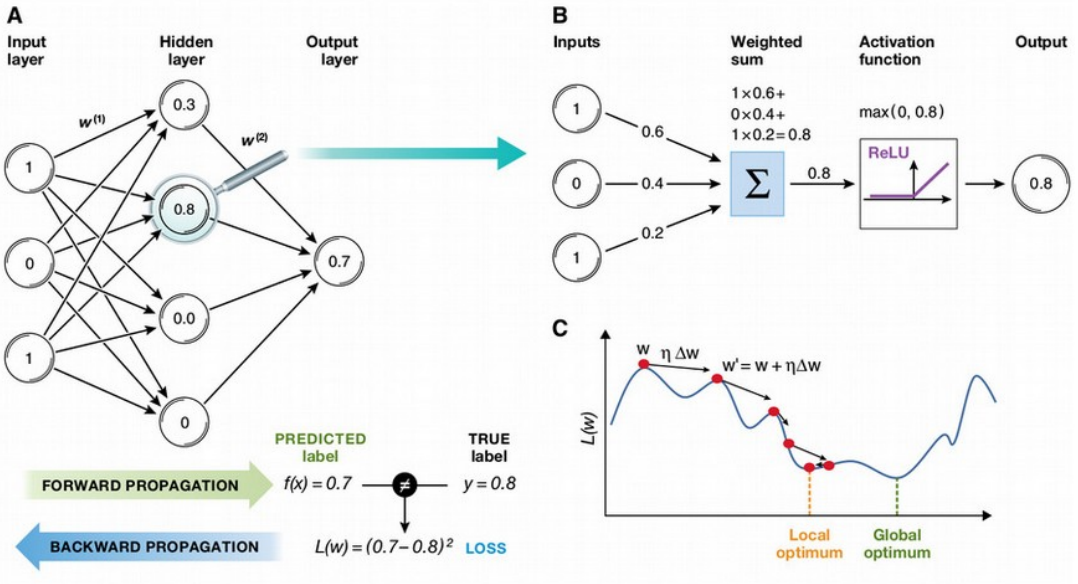


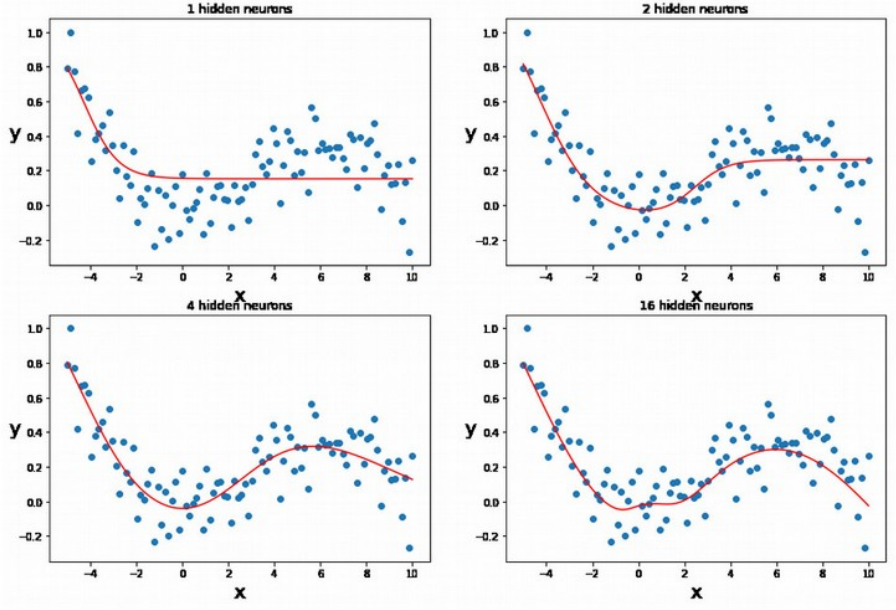
Why don’t neural networks always work?

Geoffrey Hinton: Nobel prize in physics 2024

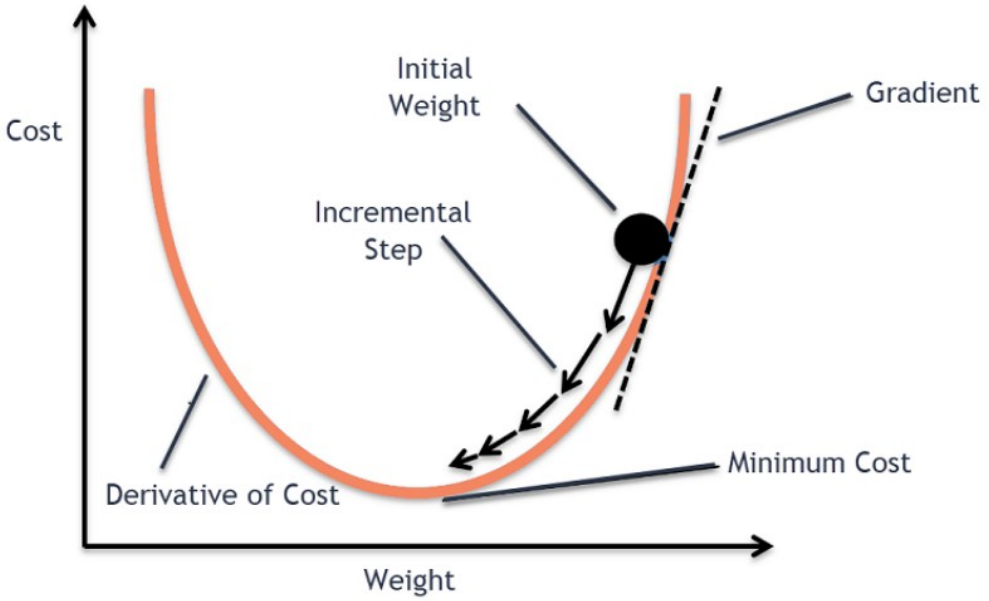
Artificial Neural Networks: general principles

- ANN: a mathematical function Y = f(X) with a special architecture
- Can be **non-linear** depending on **activation function**
- Backward propagation (**gradient descent**) for minimizing error
- Universal Approximation Theorem





Gradient descent: backward propagation



$$\hat{\alpha}, \hat{\beta} = \operatorname{argmin} \mathrm{E}(\alpha, \beta)$$

$$\frac{\partial E(\alpha, \beta)}{\partial \alpha} = -\frac{2}{n} \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)$$

$$\frac{\partial E(\alpha, \beta)}{\partial \beta} = -\frac{2}{n} \sum_{i=1}^{n} x_i (y_i - \alpha - \beta x_i)$$

Numeric implementation of gradient descent:

$$y_i = \alpha + \beta x_i + \epsilon, \ \ i = 1 \ldots n$$

$$E(\alpha, \beta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

$$\alpha_{i+1} = \alpha_i - \eta \left. \frac{\partial E(\alpha, \beta)}{\partial \alpha} \right|_{\alpha=\alpha_i, \beta=\beta_i}$$

$$\beta_{i+1} = \beta_i - \eta \left. \frac{\partial E(\alpha, \beta)}{\partial \beta} \right|_{\alpha=\alpha_i, \beta=\beta_i}$$

Coding gradient descent from scratch in R

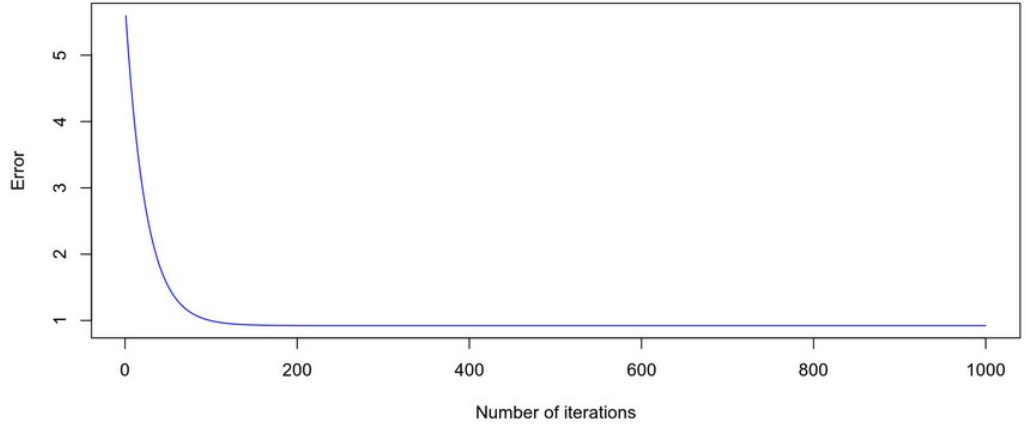
```r
n <- 100 # sample size
x <- rnorm(n) # simulated expanatory variable
y <- 3 + 2 * x + rnorm(n) # simulated response variable
summary(lm(y ~ x))
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9073 -0.6835 -0.0875  0.5806  3.2904

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.89720    0.09755   29.70   <2e-16 ***
x            1.94753    0.10688   18.22   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9707 on 98 degrees of freedom
Multiple R-squared:  0.7721,    Adjusted R-squared:  0.7698
F-statistic:   332 on 1 and 98 DF,  p-value: < 2.2e-16
```

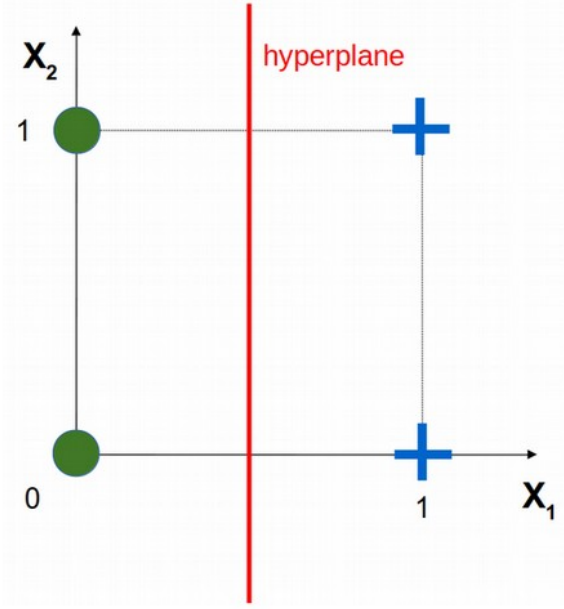
Let us now reconstruct the intercept and slope from gradient descent

```r
alpha <- vector(); beta <- vector()
E <- vector(); dEdalpha <- vector(); dEdbeta <- vector()
eta <- 0.01; alpha[1] <- 1; beta[1] <- 1 # initialize alpha and beta
for(i in 1:1000)
{
  E[i] <- (1/n) * sum((y - alpha[i] - beta[i] * x)^2)
  dEdalpha[i] <- - sum(2 * (y - alpha[i] - beta[i] * x)) / n
  dEdbeta[i] <- - sum(2 * x * (y - alpha[i] - beta[i] * x)) / n

  alpha[i+1] <- alpha[i] - eta * dEdalpha[i]
  beta[i+1] <- beta[i] - eta * dEdbeta[i]
}
print(paste0("alpha = ", tail(alpha, 1),", beta = ", tail(beta, 1)))
```

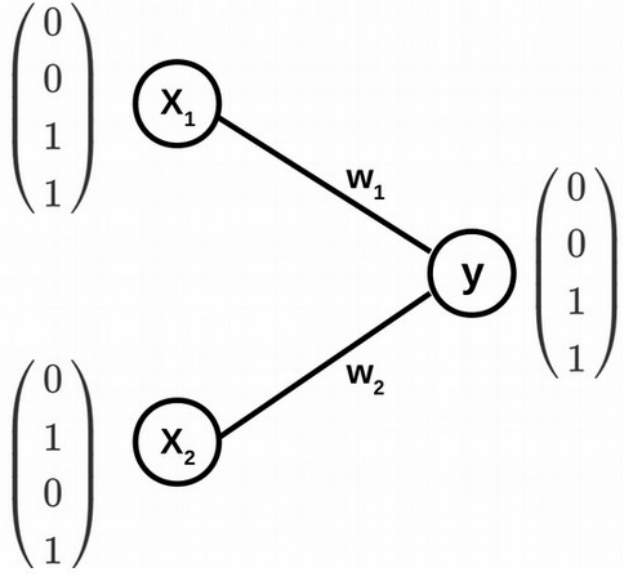
```
[1] "alpha = 2.89719694937354, beta = 1.94752837381973"
```



ANN from scratch: problem formulation



X_1	X_2	d (true) y (pred)
0	0	0 - circle
0	1	0 - circle
1	0	1 - cross
1	1	1 - cross




```r
d <- c(0, 0, 1, 1)  # true labels
x1 <- c(0, 0, 1, 1) # input variable x1
x2 <- c(0, 1, 0, 1) # input variable x2

data.frame(x1 = x1, x2 = x2, d = d)
```

x1	x2	d
0	0	0
0	1	0
1	0	1
1	1	1

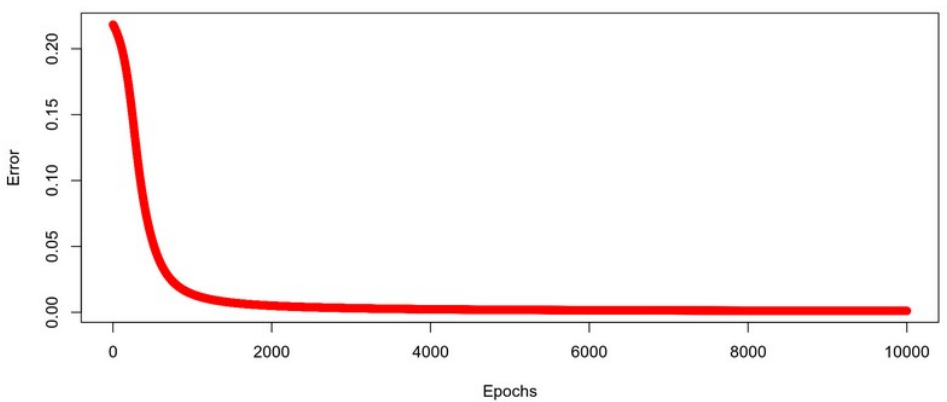
$$y(w_1, w_2) = \phi(w_1 x_1 + w_2 x_2)$$

$$\phi(s) = \frac{1}{1 + e^{-s}} - \text{sigmoid}$$

$$\phi'(s) = \phi(s)\,(1 - \phi(s))$$

Coding ANN from scratch in R

```r
phi <- function(x){return(1/(1 + exp(-x)))} # activation function

mu <- 0.1; N_epochs <- 10000
w1 <- 0.1; w2 <- 0.5; E <- vector()
for(epochs in 1:N_epochs)
{
  #Forward propagation
  y <- phi(w1 * x1 + w2 * x2 - 3) # we use a fixed bias -3

  #Backward propagation
  E[epochs] <- (1 / (2 * length(d))) * sum((d - y)^2)
  dE_dw1 <- - (1 / length(d)) * sum((d - y) * y * (1 - y) * x1)
  dE_dw2 <- - (1 / length(d)) * sum((d - y) * y * (1 - y) * x2)
  w1 <- w1 - mu * dE_dw1
  w2 <- w2 - mu * dE_dw2
}
plot(E ~ seq(1:N_epochs), xlab="Epochs", ylab="Error", col="red")
```



$$E(w_1, w_2) = \frac{1}{2N} \sum_{i=1}^{N} (d_i - y_i(w_1, w_2))^2$$

$$w_{1,2} = w_{1,2} - \mu \frac{\partial E(w_1, w_2)}{\partial w_{1,2}}$$

$$\frac{\partial E}{\partial w_1} = -\frac{1}{N} \sum_{i=1}^{N} (d_i - y_i) * y_i * (1 - y_i) * x_{1i}$$

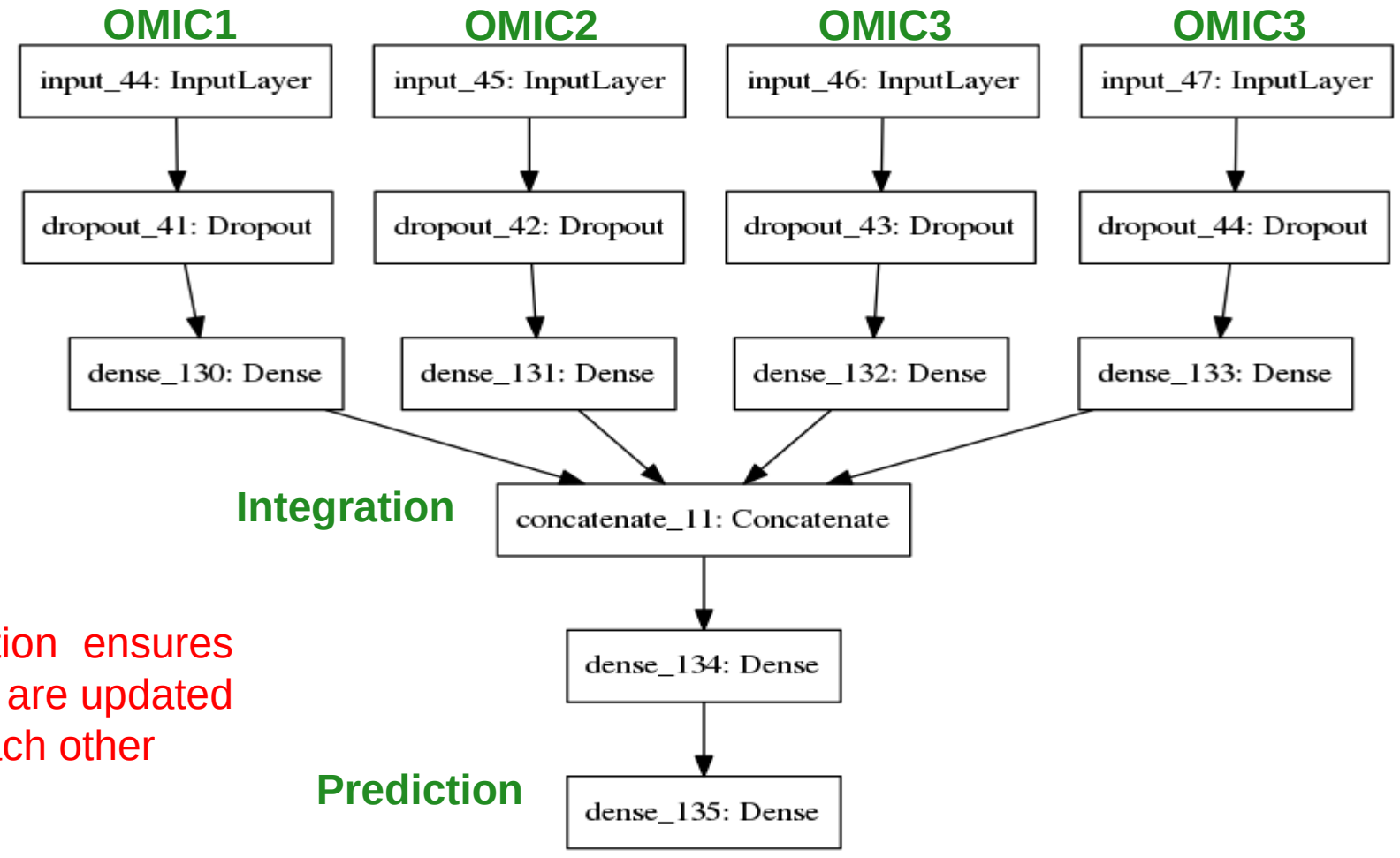
$$\frac{\partial E}{\partial w_2} = -\frac{1}{N} \sum_{i=1}^{N} (d_i - y_i) * y_i * (1 - y_i) * x_{2i}$$

```r
y
```

```
[1] 0.04742587 0.05752359 0.95730271 0.96489475
```

We nearly reconstruct true labels d = (0, 0, 1, 1)

Supervised deep learning Omics integration



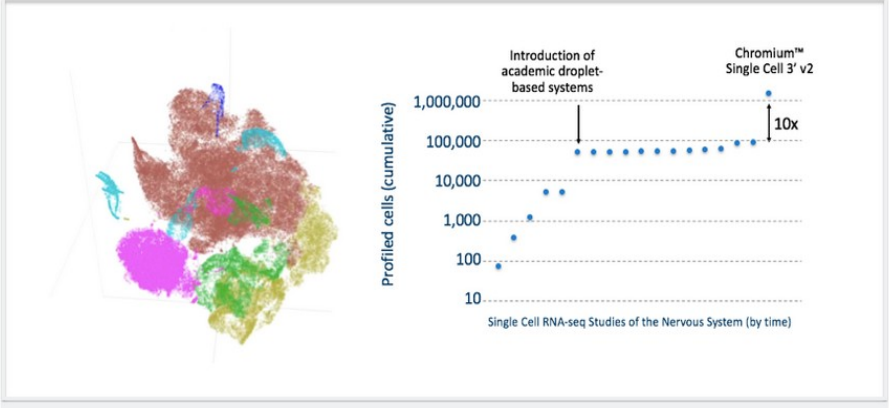
Back propagation ensures that all weights are updated in context of each other

Big Data in single cell genomics

CAREERS BLOG 10X UNIVERSITY

10X GENOMICS SOLUTIONS & PRODUCTS RESEARCH & APPLICATIONS EDUCATION & RESOURC

« Back to Blog ‹ Newer Article Older Article ›



Our 1.3 million single cell dataset is ready to download

0 KUDOS

POSTED BY: grace-10x, on Feb 21, 2017 at 2:28 PM

At ASHG last year, we announced our 1.3 Million Brain Cell Dataset, which is, to date, the largest dataset published in the single cell RNA-sequencing (scRNA-seq) field. Using the Chromium™ Single Cell 3' Solution (v2 Chemistry), we were able to sequence and profile 1,308,421 individual cells from embryonic mice brains. Read more in our application note Transcriptional Profiling of 1.3 Million Brain Cells with the Chromium™ Single Cell 3' Solution.

Available online at www.sciencedirect.com

ScienceDirect

Current Opinion in Systems Biology

Single cells make big data: New challenges and opportunities in transcriptomics

Philipp Angerer[1], Lukas Simon[1], Sophie Tritschler[1], F. Alexander Wolf[1], David Fischer[1] and Fabian J. Theis[1,2]

Abstract

Recent technological advances have enabled unprecedented insight into transcriptomics at the level of single cells. Single cell transcriptomics enables the measurement of transcriptomic information of thousands of single cells in a single experiment. The volume and complexity of resulting data make it a paradigm of big data. Consequently, the field is presented with new scientific and, in particular, analytical challenges where currently no scalable solutions exist. At the same time, exciting opportunities arise from increased resolution of single-cell RNA sequencing data and improved statistical power of ever growing datasets. Big single cell RNA sequencing data promises valuable insights into cellular heterogeneity which may significantly improve our understanding of biology and human disease. This review focuses on single cell transcriptomics and highlights the inherent opportunities and challenges in the context of big data analytics.

Addresses

[1] Institute of Computational Biology, Helmholtz Center Munich, German Research Center for Environmental Health, Neuherberg, Germany
[2] Department of Mathematics, Technical University of Munich, Garching, Germany

Corresponding author: Theis, Fabian J. Institute of Computational Biology, Helmholtz Center Munich, German Research Center for Environmental Health, Neuherberg, Germany. (fabian.theis@helmholtz-muenchen.de)

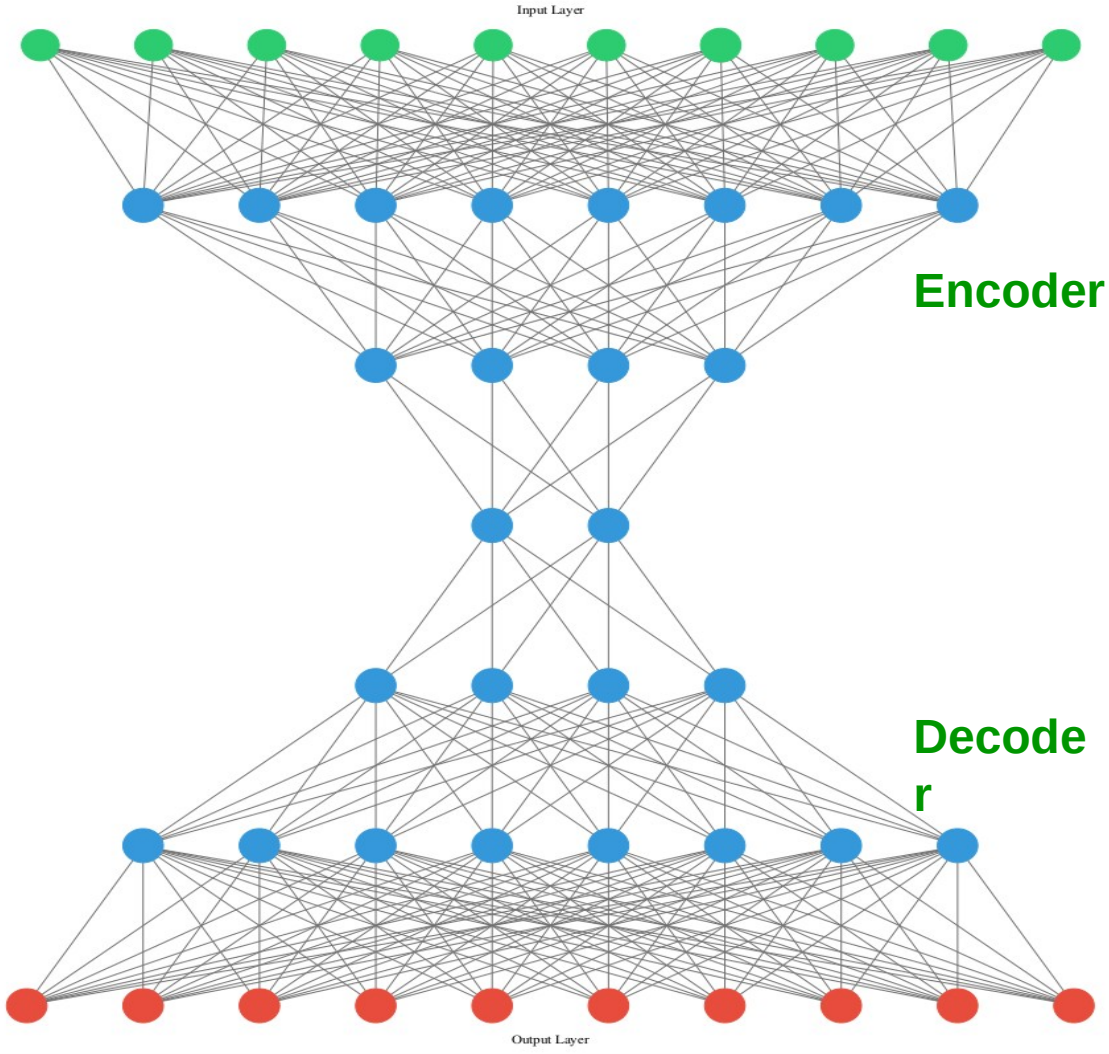
Current Opinion in Systems Biology 2017, 4:85–91

This review comes from a themed issue on **Big data acquisition and**

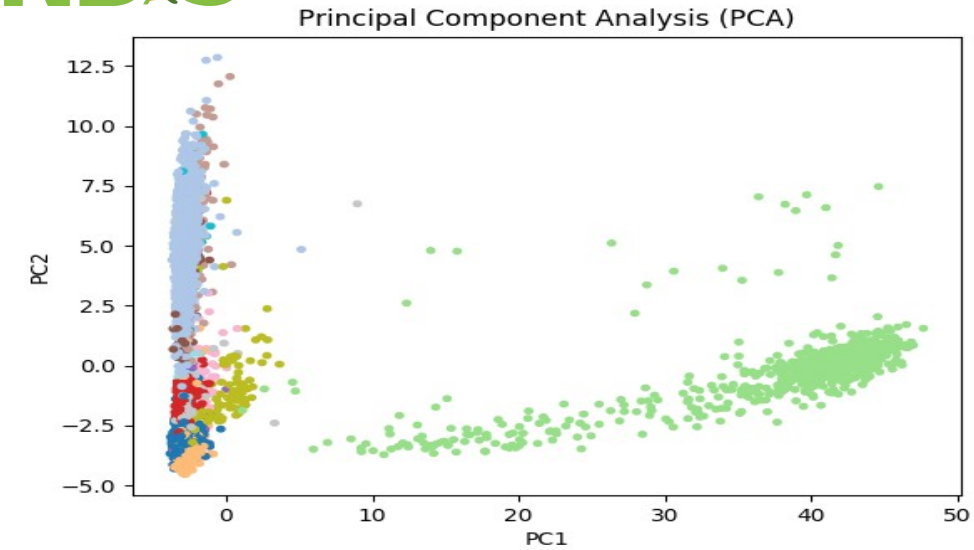
volume — the amount of data, velocity — the required processing speed, veracity — trustworthiness and availability, and variety — necessary model complexity [2]. The traditional scientific big data field is astronomy because of the huge *volume* of image data produced by telescopes with a high daily *velocity* [3]. Big data has also reached biology, mainly driven through the advent of next generation sequencing technology. For biologists, assessing *veracity* through statistical means is nothing new.

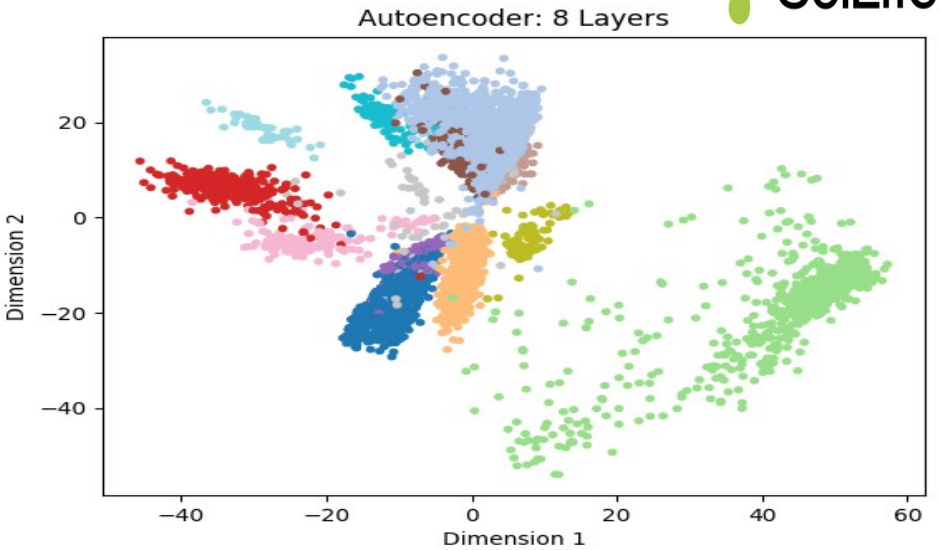
Recent technological advances now allow the profiling of single cells at a *variety* of omic layers (genomes, epigenomes, transcriptomes and proteomes) at an unprecedented level of resolution [4]. Single cell transcriptomics (SCT) entails the profiling of all messenger RNAs present in a single cell and constitutes the most widely-used sc profiling technology [4]. Unlike bulk RNA-seq profiling where sequencing libraries are generated from thousands of cells, scRNA-seq technologies isolate single cells and generate cell-specific sequencing libraries (e.g. Fluidigm [5]) mark RNA content with a cell-specific molecular barcode [6–9]. Both approaches generate gene expression estimates at the single cell level [10]. SCTenables, for the first time, the measurement of the transcriptomic information of thousands, and up to millions of single cells, in a single experiment [7]. The complexity of SCT data coupled with the massive volume inherent to next generation sequencing data makes it a paradigm of big data.

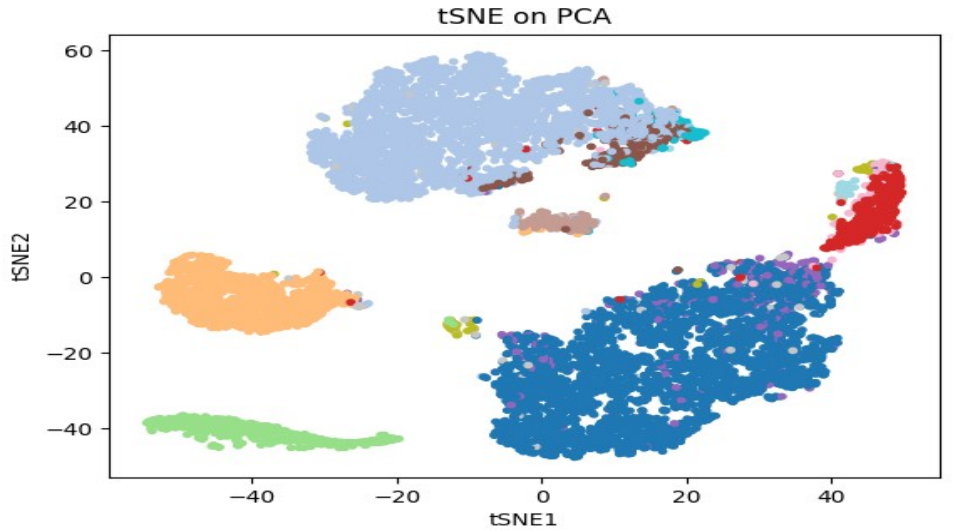
Autoencoder: unsupervised neural network

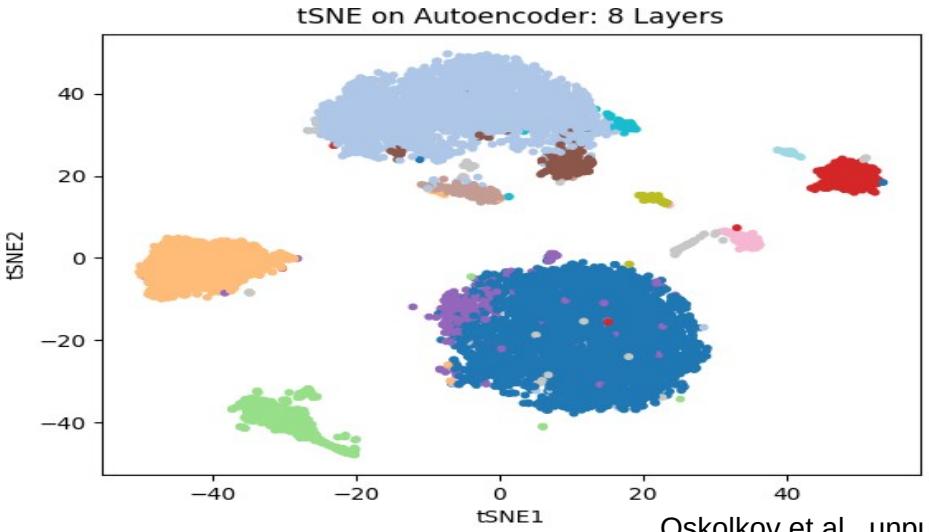


Autoencoder dimension reduction: 8617 PBMC cells



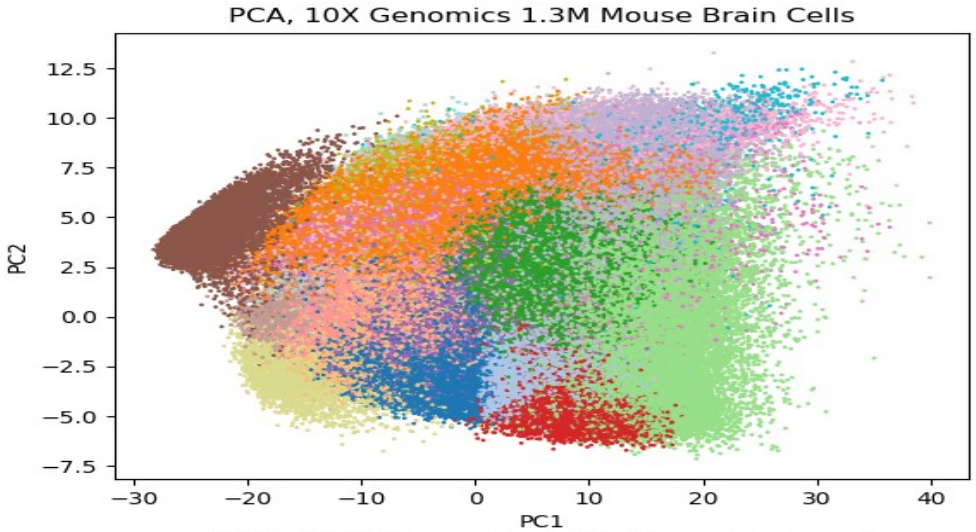


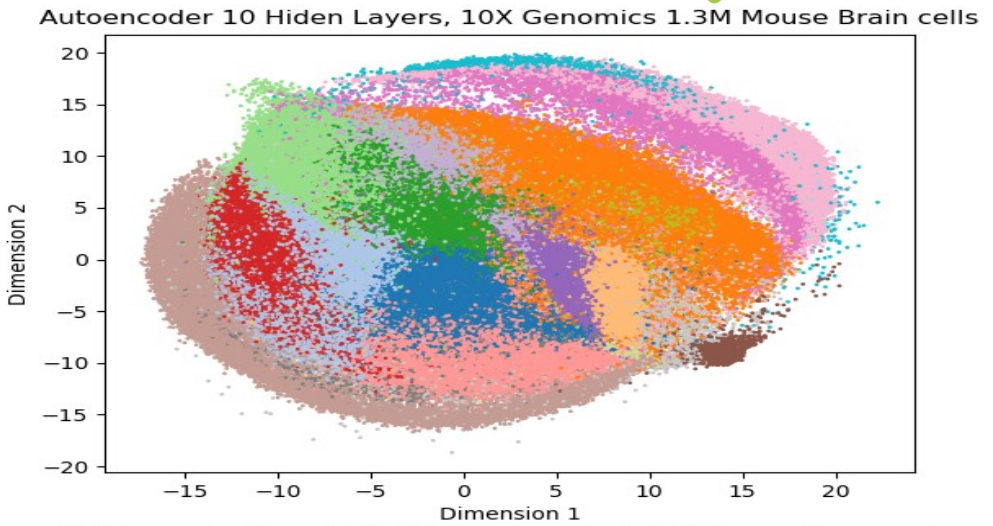


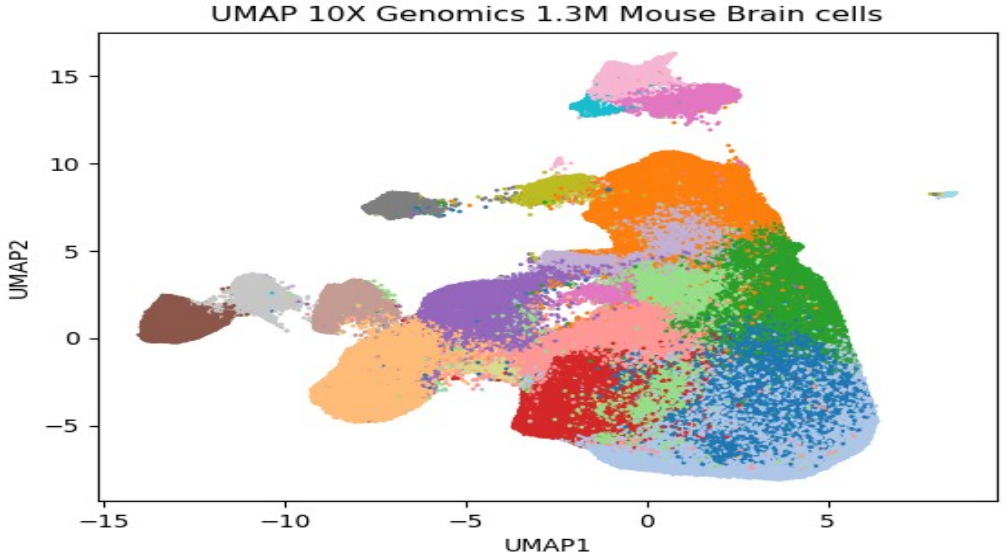


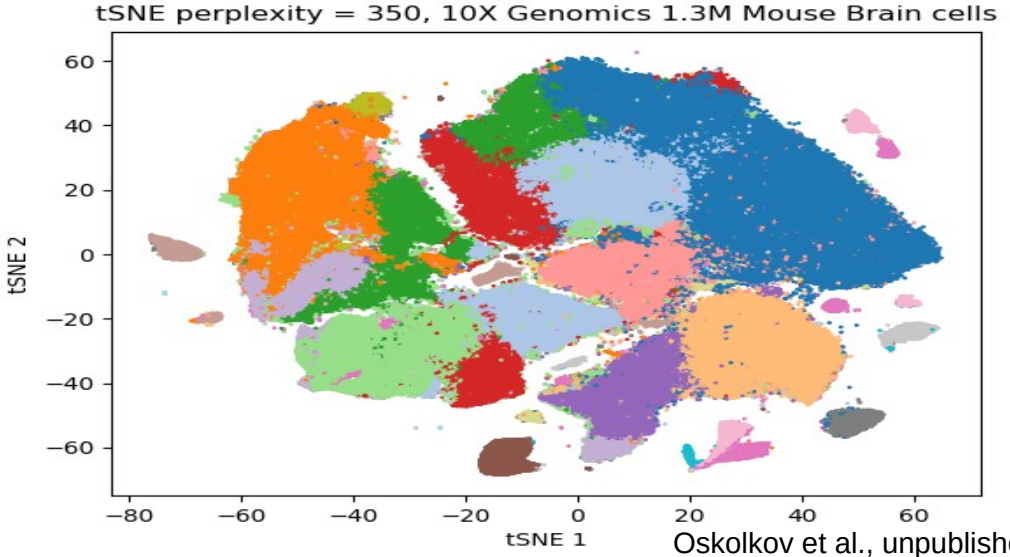
Oskolkov et al., unpublished

10X Genomics Mouse Brain: scRNAseq, 1.3M cells



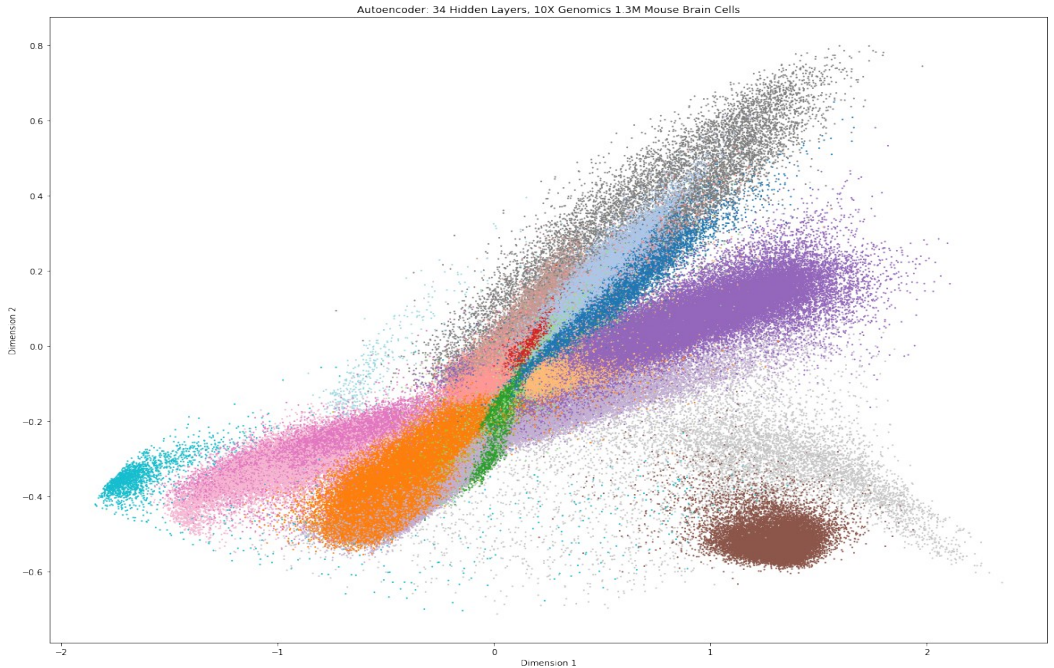




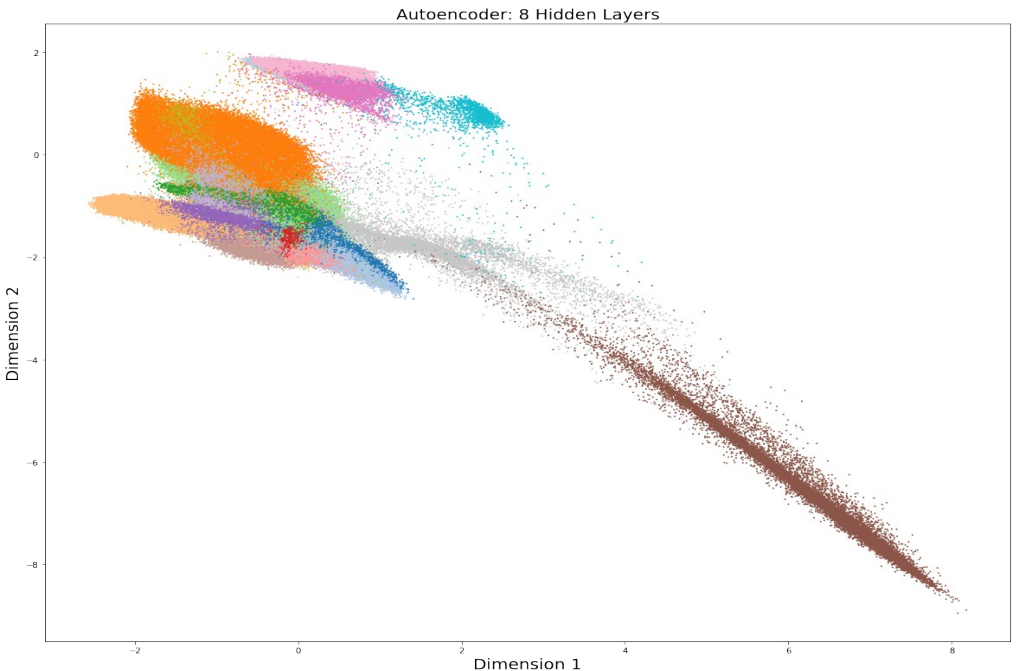


Oskolkov et al., unpublished

10X Genomics mouse brain: scRNAseq, 1.3M cells



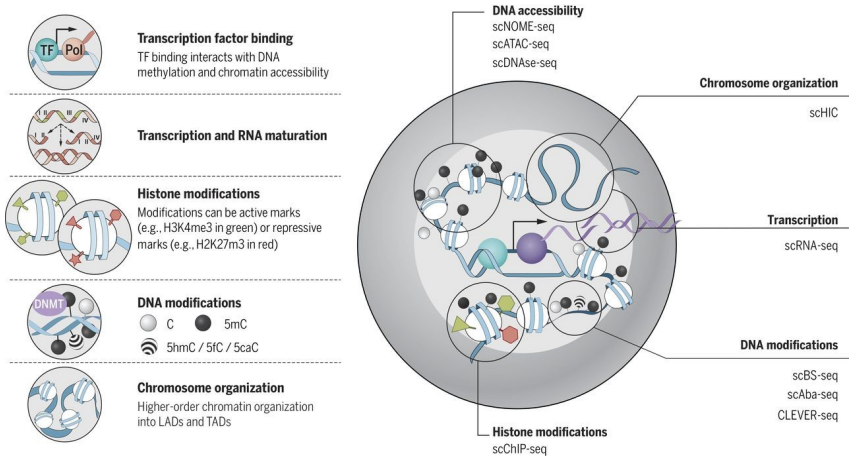
Autoencoders can be promising for non-linear data pre-processing, the bottleneck can potentially be fed to tSNE / UMAP



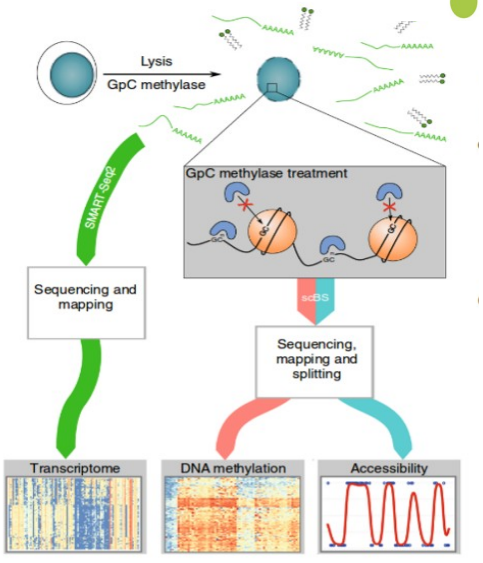
Autoencoders themselves are perhaps not optimal for visualization of scOmics

Oskolkov et al., unpublished

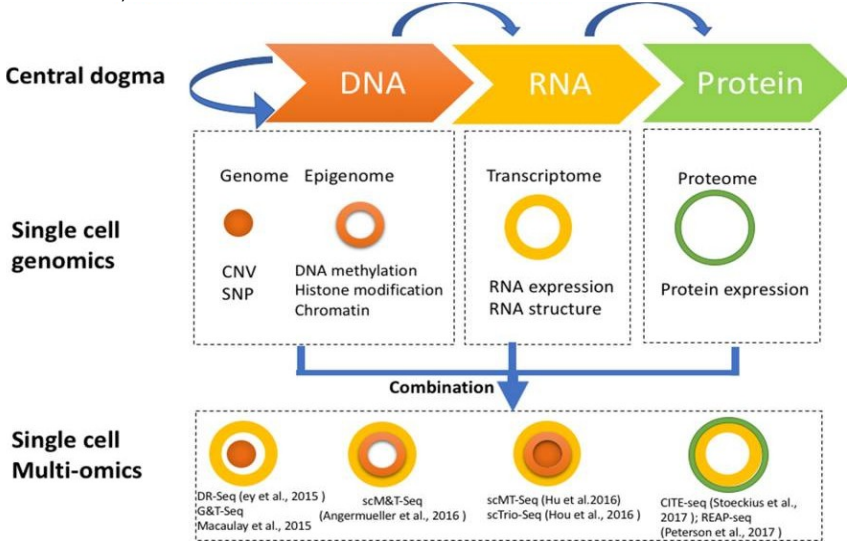
Multi-Omics in Single Cell Genomics



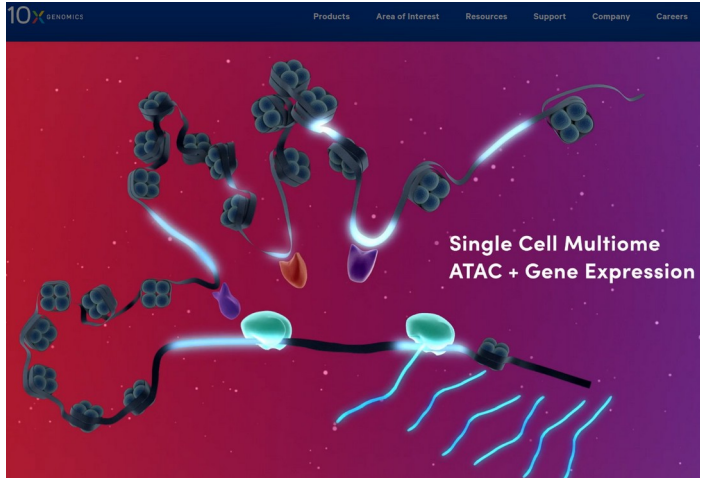
Kelsey et al., 2017, Science 358, 69-75



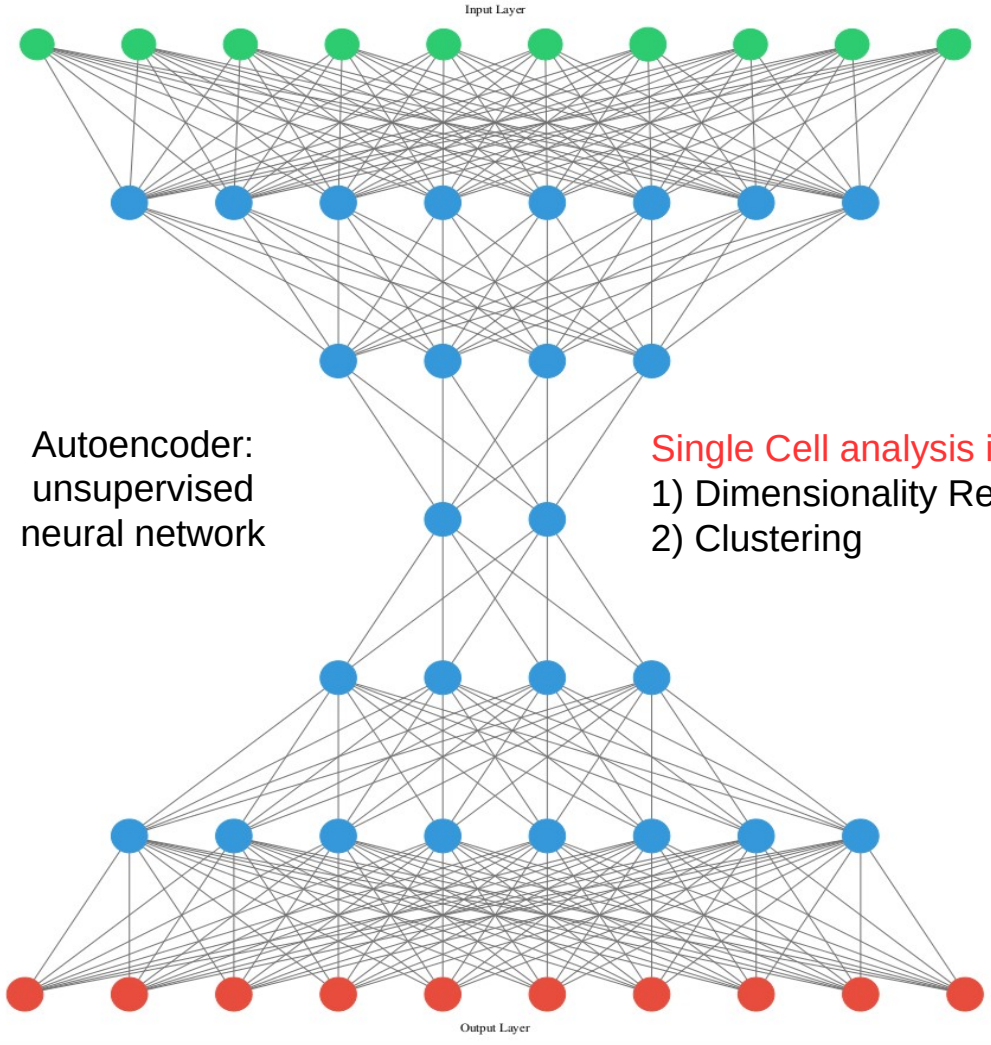
Clark et al., 2018, Nature Communications 9, 781



Hu et al., 2018, Frontier in Cell and Developmental Biology 6, 1-13

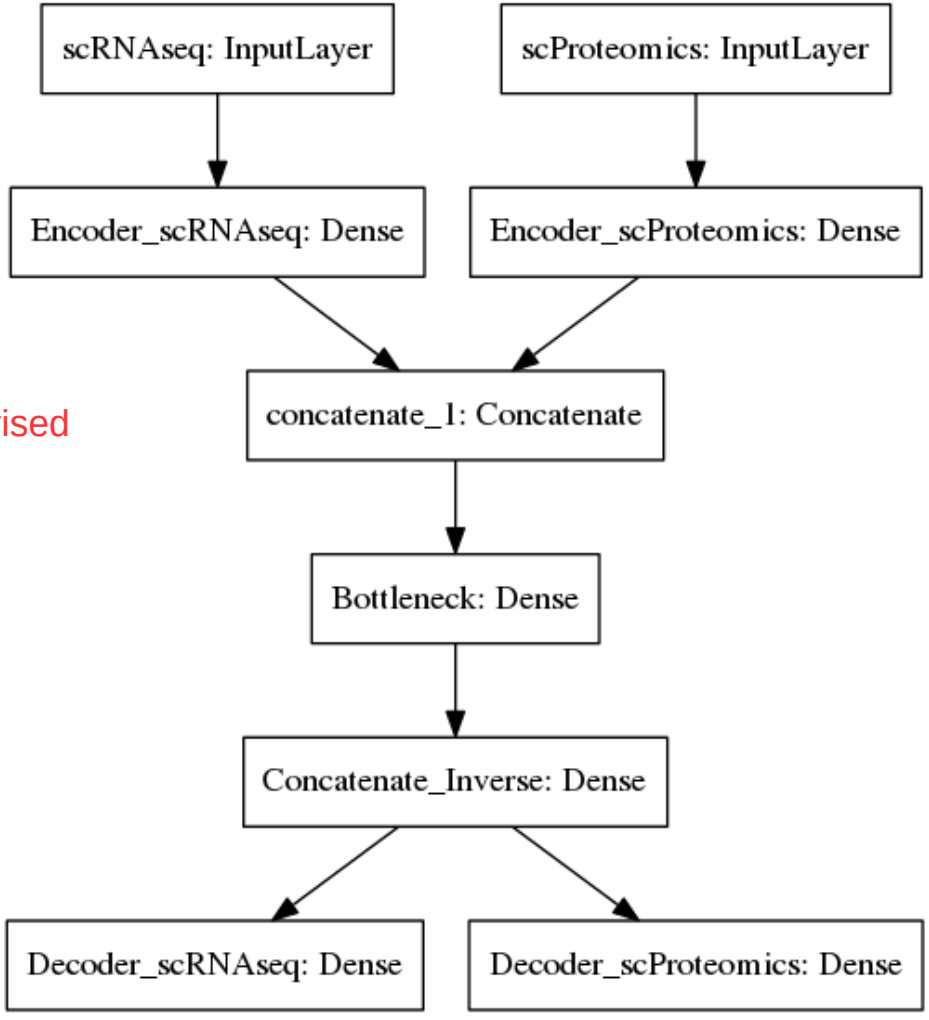


Autoencoder for data integration

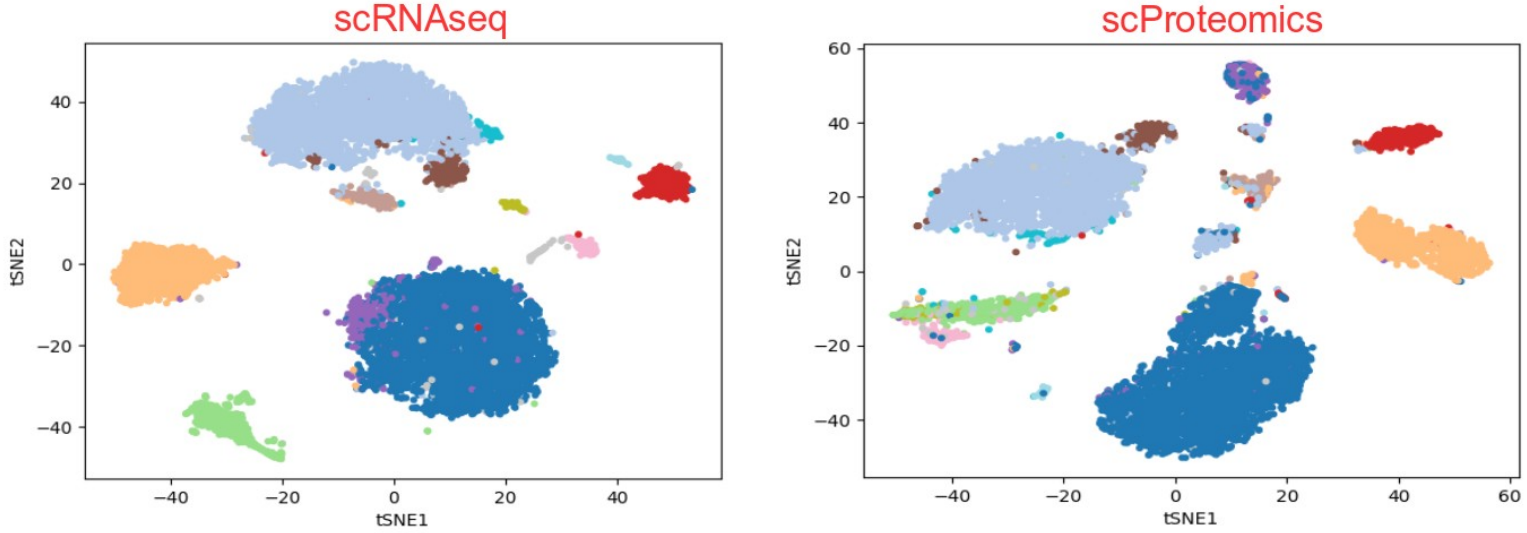


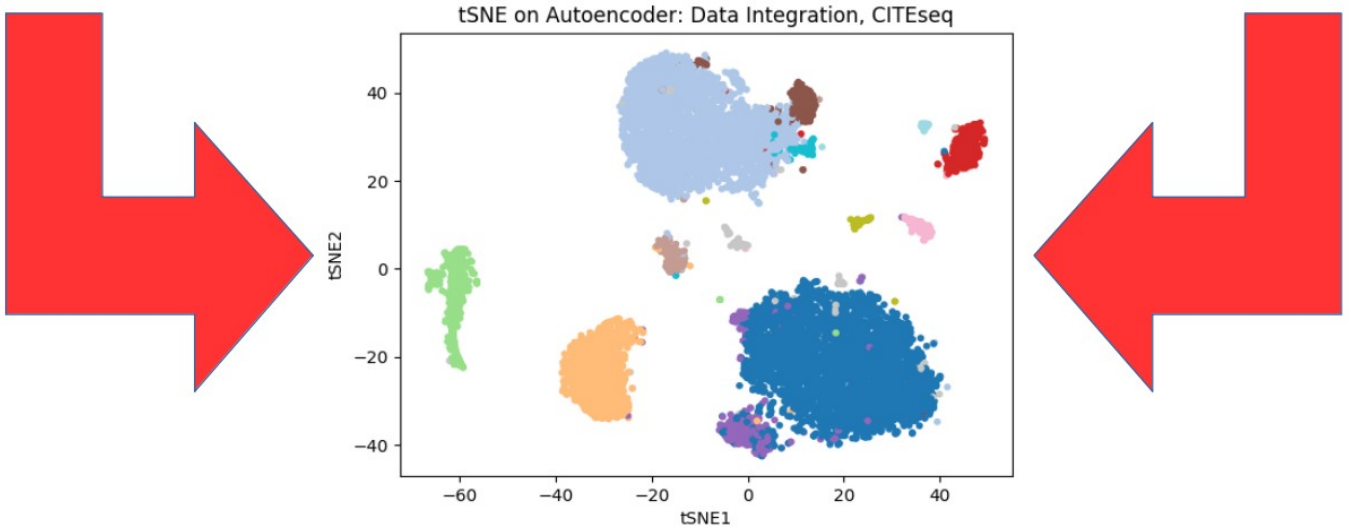
Autoencoder:
unsupervised
neural network

Single Cell analysis is unsupervised
1) Dimensionality Reduction
2) Clustering

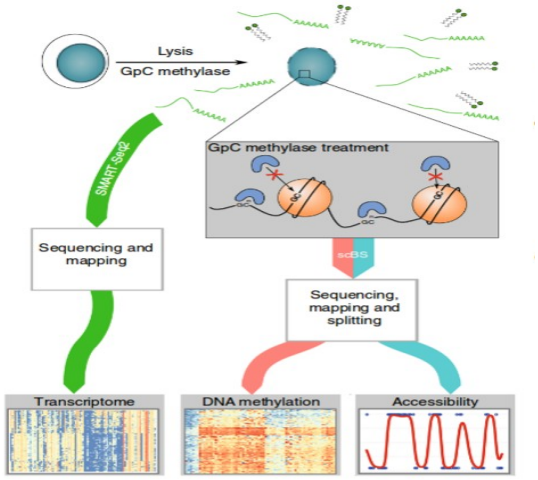


Autoencoder for data integration: CITEseq

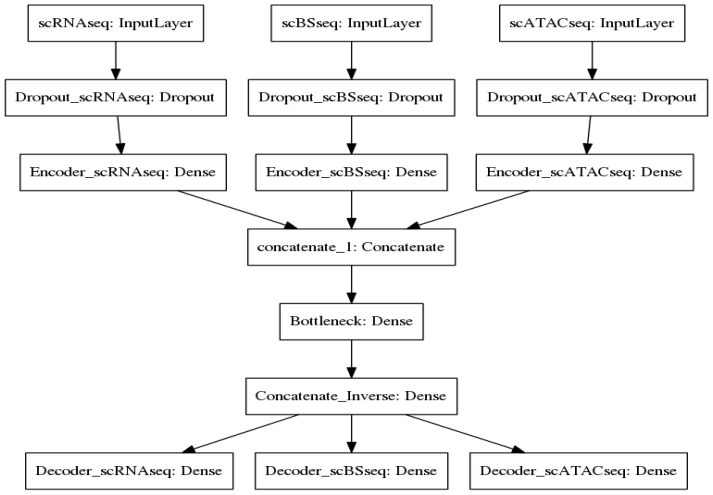


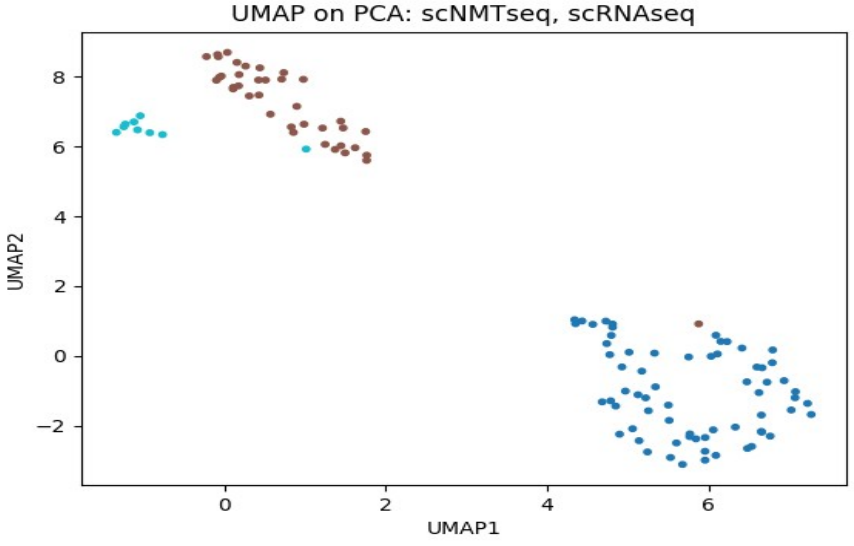


scNMTseq: scRNAseq + scBSseq + scATACseq

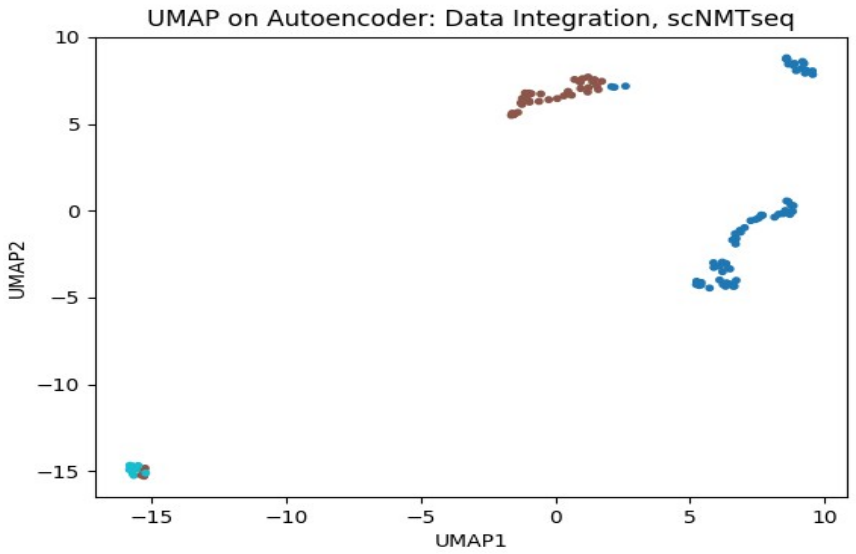


scNMTseq: Clark et al., 2018, Nature Communications 9, 781

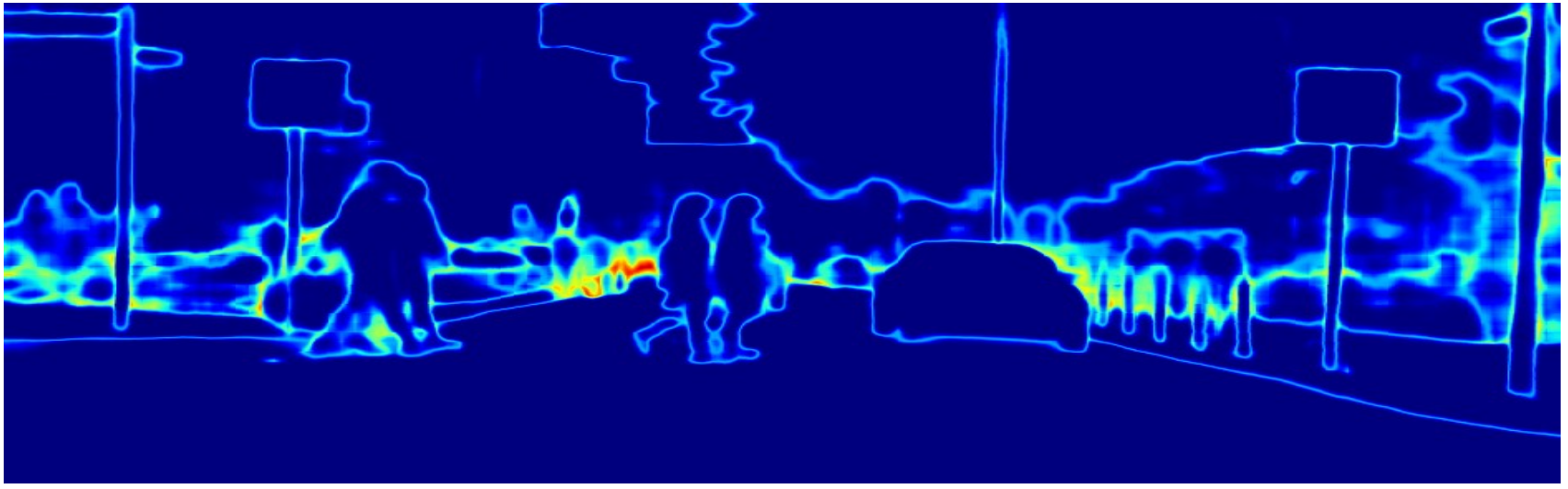




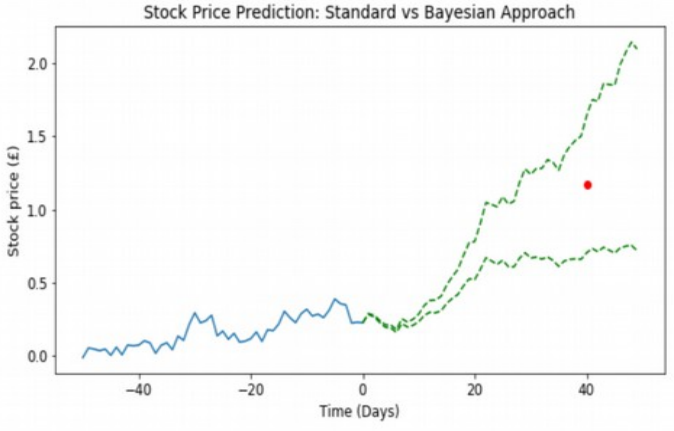
From Single
To
multi-Omics

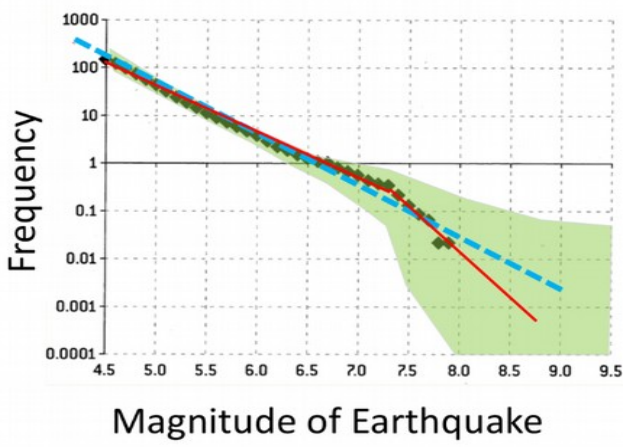


When Deep Learning is not good enough



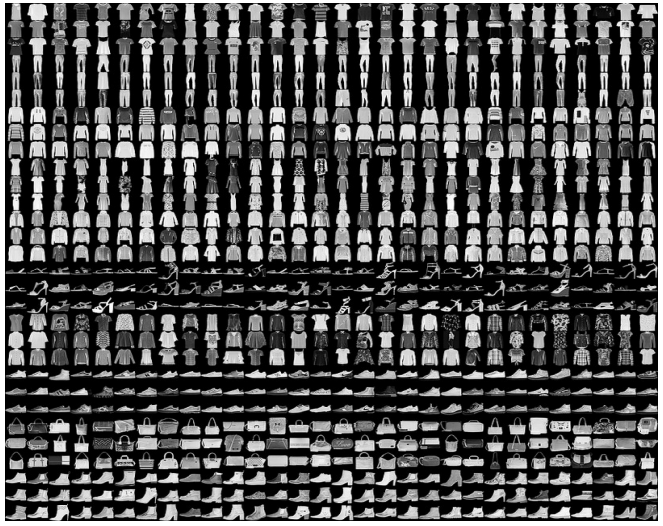
Intelligence is to know how much you do not know





Frequentist image recognition

Fashion MNIST



```
In [24]: # normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0

In [25]: # one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
print(num_classes)
10

In [27]: # Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(1, 28, 28), padding='same', activation='relu',
                 kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), padding='same', activation='relu',
                 kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
```

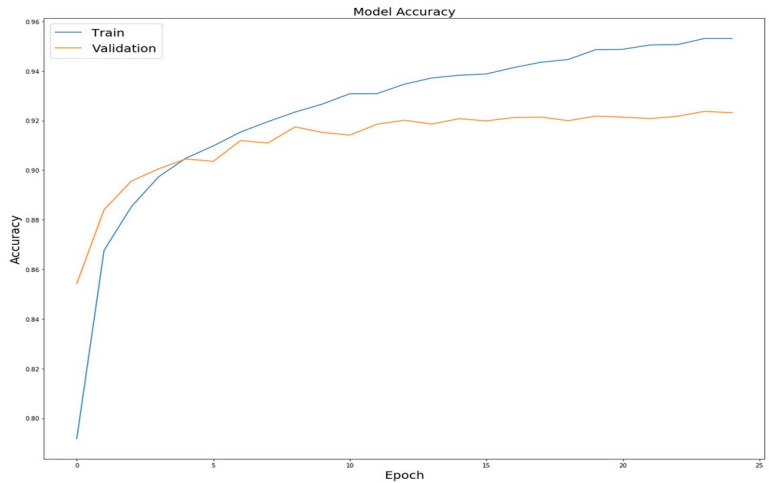
Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 32, 28, 28)	320
dropout_7 (Dropout)	(None, 32, 28, 28)	0
conv2d_9 (Conv2D)	(None, 32, 28, 28)	9248
max_pooling2d_4 (MaxPooling2	(None, 32, 14, 14)	0
flatten_4 (Flatten)	(None, 6272)	0
dense_7 (Dense)	(None, 512)	3211776
dropout_8 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 10)	5130

Total params: 3,226,474
Trainable params: 3,226,474
Non-trainable params: 0

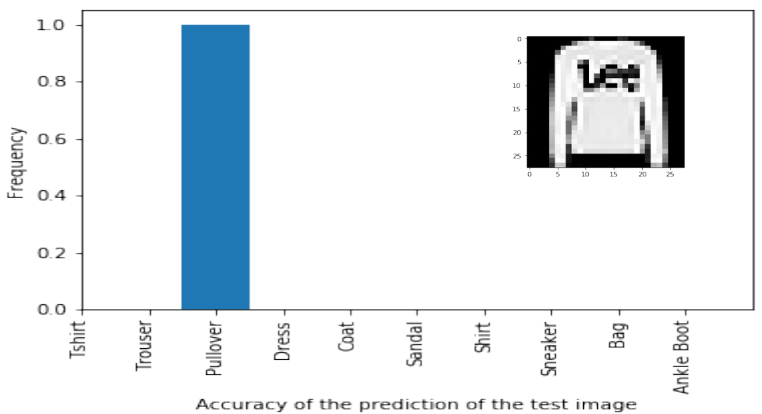
None

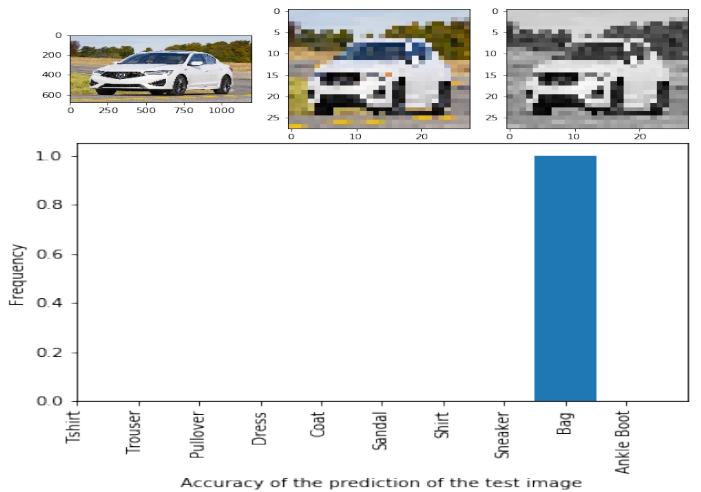
```
In [28]: # Fit the model
#model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
history = model.fit(X_train, y_train, epochs = epochs, verbose = 1, validation_split = 0.25,
                    batch_size = 32, shuffle = True)
```

```
Train on 45000 samples, validate on 15000 samples
Epoch 1/25
45000/45000 [==============================] - 1158s 26ms/step - loss: 0.5762 - acc: 0.7917 - val_loss: 0.39
73 - val_acc: 0.8542
Epoch 2/25
45000/45000 [==============================] - 1124s 25ms/step - loss: 0.3643 - acc: 0.8676 - val_loss: 0.31
27 - val_acc: 0.8841
Epoch 3/25
45000/45000 [==============================] - 1158s 26ms/step - loss: 0.3129 - acc: 0.8853 - val_loss: 0.28
25 - val_acc: 0.8956
Epoch 4/25
45000/45000 [==============================] - 1609s 36ms/step - loss: 0.2813 - acc: 0.8973 - val_loss: 0.27
27 - val_acc: 0.9005
Epoch 5/25
45000/45000 [==============================] - 902s 20ms/step - loss: 0.2618 - acc: 0.9048 - val_loss: 0.258
8 - val_acc: 0.9045
Epoch 6/25
45000/45000 [==============================] - 936s 21ms/step - loss: 0.2451 - acc: 0.9098 - val_loss: 0.256
4 - val_acc: 0.9035
Epoch 7/25
```



Prediction



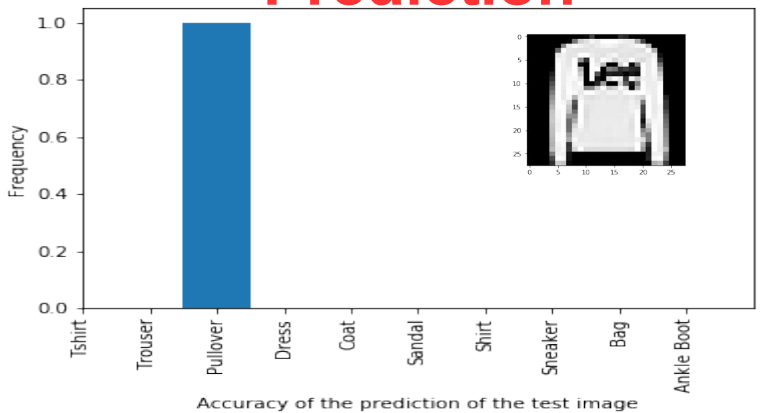


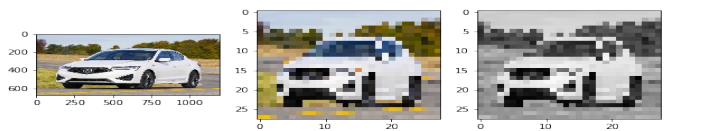
Bayesian image recognition

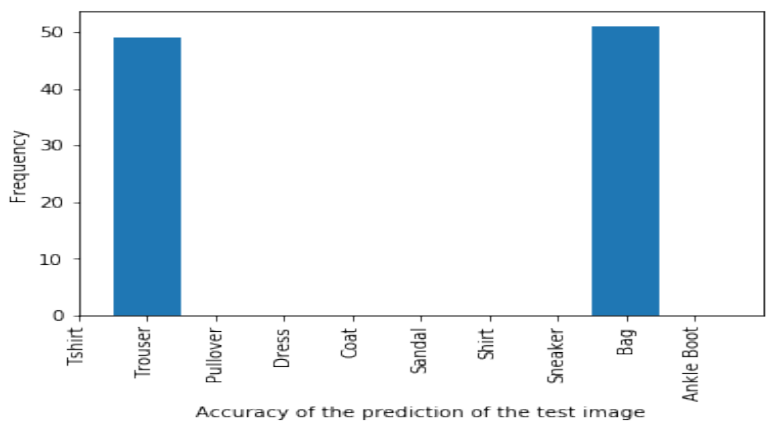
PyMC3, Edward, TensorFlow Probability

```
In [8]: x_train = x_train.reshape((x_train.shape[0],D))
        x_test = x_test.reshape((x_test.shape[0],D))
        print(x_train.shape)
        print(x_test.shape)

        (60000, 784)
        (10000, 784)

In [9]: from keras.utils import to_categorical
        y_train = to_categorical(y_train)
        y_test = to_categorical(y_test)
        print(y_train.shape)
        print(y_test.shape)

        (60000, 10)
        (10000, 10)

In [10]: ed.set_seed(314159)
         N = 100      # number of images in a minibatch.
         D = D        # number of features.
         K = 10       # number of classes.

         # Create a placeholder to hold the data (in minibatches) in a TensorFlow graph.
         x = tf.placeholder(tf.float32, [None, D])
         # Normal(0,1) priors for the variables. Note that the syntax assumes TensorFlow 1.1.
         w = Normal(loc=tf.zeros([D, K]), scale=tf.ones([D, K]))
         b = Normal(loc=tf.zeros(K), scale=tf.ones(K))
         # Categorical likelihood for classication.
         y = Categorical(tf.matmul(x, w) + b)

In [11]: # Contruct the q(w) and q(b). in this case we assume Normal distributions.
         qw = Normal(loc=tf.Variable(tf.random_normal([D, K])),
                     scale=tf.nn.softplus(tf.Variable(tf.random_normal([D, K]))))
         qb = Normal(loc=tf.Variable(tf.random_normal([K])),
                     scale=tf.nn.softplus(tf.Variable(tf.random_normal([K]))))

In [12]: def generator(arrays, batch_size = N):
             starts = [0] * len(arrays)  # pointers to where we are in iteration
             while True:
                 batches = []
                 for i, array in enumerate(arrays):
                     start = starts[i]
                     stop = start + batch_size
                     diff = stop - array.shape[0]
                     if diff <= 0:
                         batch = array[start:stop]
                         starts[i] += batch_size
                     else:
                         batch = np.concatenate((array[start:], array[:diff]))
                         starts[i] = diff
                     batches.append(batch)
                 yield batches
         cifar10 = generator([x_train, y_train], N)

In [13]: # We use a placeholder for the labels in anticipation of the traning data.
         y_ph = tf.placeholder(tf.int32, [N])
         # Define the VI inference technique, ie. minimise the KL divergence between q and p.
         inference = ed.KLqp({w: qw, b: qb}, data={y: y_ph})
         # Initialse the infernce variables
         inference.initialize(n_iter=50000, n_print=100, scale={y: float(x_train.shape[0]) / N})
         # We will use an interactive session.
         sess = tf.InteractiveSession()
         # Initialise all the vairables in the session.
         tf.global_variables_initializer().run()
         # Let the training begin. We load the data in minibatches and update the VI infernce using each new batch.
         for _ in range(inference.n_iter):
             X_batch, Y_batch = next(cifar10)
             #X_batch = X_batch.reshape(N, -1)
             # TensorFlow method gives the label data in a one hot vetor format. We convert that into a single label.
             Y_batch = np.argmax(Y_batch, axis=1)
             info_dict = inference.update(feed_dict={x: X_batch, y_ph: Y_batch})
             inference.print_progress(info_dict)
         50000/50000 [100%] ██████████████████████████████ Elapsed: 221s | Loss: 85453.266

In [14]: # Generate samples the posterior and store them.
         n_samples = 100
         prob_lst = []
         samples = []
         w_samples = []
         b_samples = []
         for _ in range(n_samples):
             w_samp = qw.sample()
             b_samp = qb.sample()
             w_samples.append(w_samp)
             b_samples.append(b_samp)
             # Also compute the probabiliy of each class for each (w,b) sample.
             prob = tf.nn.softmax(tf.matmul(x_test, w_samp) + b_samp)
             prob_lst.append(prob.eval())
             sample = tf.concat([tf.reshape(w_samp,[-1]),b_samp],0)
             samples.append(sample.eval())

In [15]: # Compute the accuracy of the model.
         # For each sample we compute the predicted class and compare with the test labels.
         # Predicted class is defined as the one which as maximum probability.
         # We perform this test for each (w,b) in the posterior giving us a set of accuracies
         # Finally we make a histogram of accuracies for the test data.
         accy_test = []
         for prob in prob_lst:
             y_trn_prd = np.argmax(prob, axis=1).astype(np.float32)
             acc = (y_trn_prd == np.argmax(y_test, axis=1)).mean()*100
             accy_test.append(acc)

         plt.hist(accy_test)
         plt.title("Histogram of prediction accuracies in the CIFAR10 test data")
         plt.xlabel("Accuracy")
         plt.ylabel("Frequency")
```

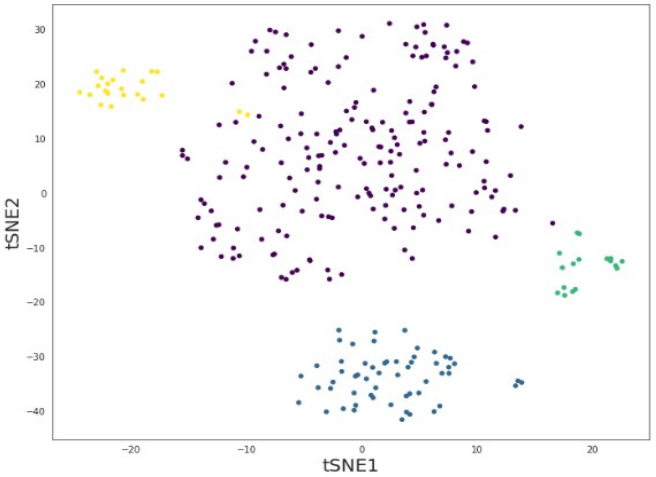
Prediction

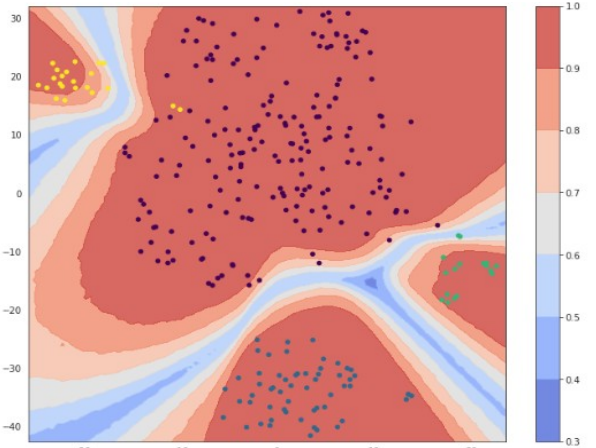


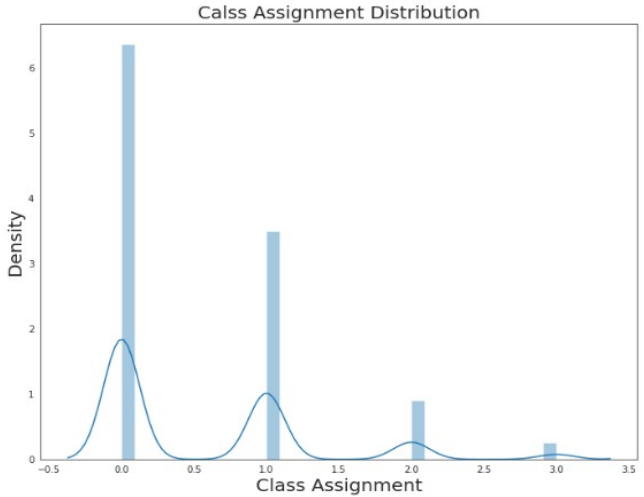


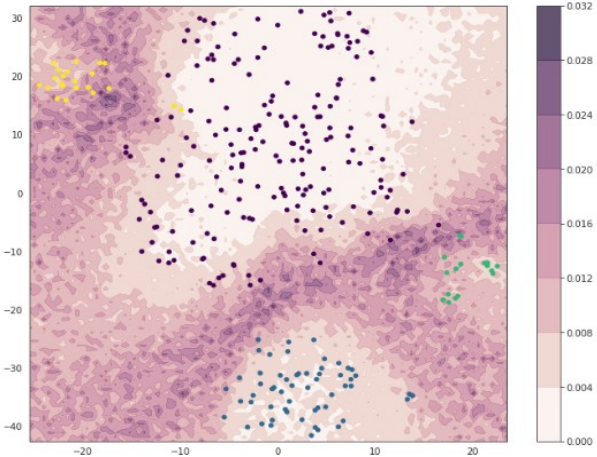


Bayesian deep learning for single cell









Bartoschek, Oskolkov et al., Nature Communications 2018

1) handy for unseen (during training) data, 2) uncertainties are crucial for clinical diagnostics

AI applications for single cell biology

nature methods
ARTICLES
https://doi.org/10.1038/s41592-018-0229-2

Deep generative modeling for single-cell transcriptomics

Romain Lopez[1], Jeffrey Regier[1], Michael B. Cole[2], Michael I. Jordan[1,3] and Nir Yosef[1,4,5*]

Single-cell transcriptome measurements can reveal unexplored biological diversity, but they suffer from technical noise and bias that must be modeled to account for the resulting uncertainty in downstream analyses. Here we introduce single-cell variational inference (scVI), a ready-to-use scalable framework for the probabilistic representation and analysis of gene expression in single cells (https://github.com/YosefLab/scVI). scVI uses stochastic optimization and deep neural networks to aggregate information across similar cells and genes and to approximate the distributions that underlie observed expression values, while accounting for batch effects and limited sensitivity. We used scVI for a range of fundamental analysis tasks including batch correction, visualization, clustering, and differential expression, and achieved high accuracy for each task.

nature methods

Article
https://doi.org/10.1038/s41592-024-02201-0

scGPT: toward building a foundation model for single-cell multi-omics using generative AI

Received: 12 July 2023
Accepted: 30 January 2024
Published online: 26 February 2024
Check for updates

Haotian Cui[1,2,3,8], Chloe Wang[1,2,3,8], Hassaan Maan[1,3,4], Kuan Pang[2,3], Fengning Luo[2,3], Nan Duan[5] & Bo Wang[1,2,3,4,6,7]

Generative pretrained models have achieved remarkable success in various domains such as language and computer vision. Specifically, the

nature COMMUNICATIONS

ARTICLE

DOI: 10.1038/s41467-018-04368-5 OPEN

Interpretable dimensionality reduction of single cell transcriptome data with deep generative models

Jiarui Ding[1,2,3,4], Anne Condon[1] & Sohrab P. Shah[1,2,3,5]

Huang et al. Genome Biology (2023) 24:259
https://doi.org/10.1186/s13059-023-03100-x

Genome Biology

RESEARCH Open Access

Evaluation of deep learning-based feature selection for single-cell RNA sequencing data analysis

Hao Huang[1,2,3], Chunlei Liu[1,3], Manoj M. Wagle[1,2,3] and Pengyi Yang[1,2,3,4*]

Biomedicine & Pharmacotherapy 165 (2023) 115077

Contents lists available at ScienceDirect
Biomedicine & Pharmacotherapy
journal homepage: www.elsevier.com/locate/biopha

Review

Deep learning applications in single-cell genomics and transcriptomics data analysis

Nafiseh Erfanian[a], A. Ali Heydari[b,c], Adib Miraki Feriz[a], Pablo Iañez[d], Afshin Derakhshani[e], Mohammad Ghasemigol[f], Mohsen Farahpour[g], Seyyed Mohammad Razavi[g], Saeed Nasseri[h], Hossein Safarpour[h,*], Amirhossein Sahebkar[i,j,k,**]

Popular summary of AI for Life Sciences

towards data science

LATEST | EDITORS' PICKS | DEEP DIVES | ABOUT | CONTRIBUTE | NEWSLETTER

TAGGED IN

DI For Life Sciences

tds Towards Data Science

Your home for data science. A Medium publication sharing concepts, ideas and codes.

More information

FOLLOWERS

688K

ELSEWHERE

MORE, ON MEDIUM

DI For Life Sciences

Nikolay Oskolkov in Towards Data Science

Aug 18, 2021 · 13 min read ★

DEEP LEARNING FOR LIFE SCIENCES

Deep Learning on Human Microbiome

Infer microbial...

Read more...

133 4 responses

Nikolay Oskolkov in Towards Data Science

May 20, 2019 · 7 min read ★

Deep Learning for Life Sciences

Deep Learning for Clinical Diagnostics

Read more...

141

Nikolay Oskolkov in Towards Data Science

Aug 6, 2019 · 6 min read ★

Deep Learning for Life Sciences

Why Biology is Sceptic Towards AI

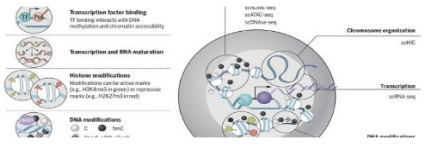
And why Precision...

Read more...

159 1 response

Nikolay Oskolkov in Towards Data Science

May 12, 2019 · 7 min read ★



Deep Learning for Life Sciences

Deep Learning for Data Integration

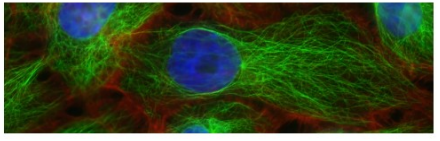
Synergistic effects...

Read more...

126 5 responses

Nikolay Oskolkov in Towards Data Science

Jul 7, 2019 · 9 min read ★



Deep Learning for Life Sciences

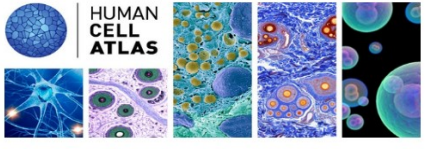
Deep Learning on Microscopy Imaging

Read more...

207 1 response

Nikolay Oskolkov in Towards Data Science

May 5, 2019 · 9 min read ★



Deep Learning for Life Sciences

Deep Learning for Single Cell Biology

Read more...

406 7 responses

Nikolay Oskolkov in Towards Data Science

Apr 28, 2019 · 6 min read ★

Deep Learning for Life Sciences

Deep Learning on Ancient DNA

Reconstructing the Human...

Read more...

381 4 responses

National Bioinformatics Infrastructure Sweden (NBIS)

Knut och Alice Wallenbergs Stiftelse

Vetenskapsrådet

Lunds Universitet