Chapter 6. Deep Learning for Genomics

고준수
리얼딥바이오
2019-08-09

ARONTIER

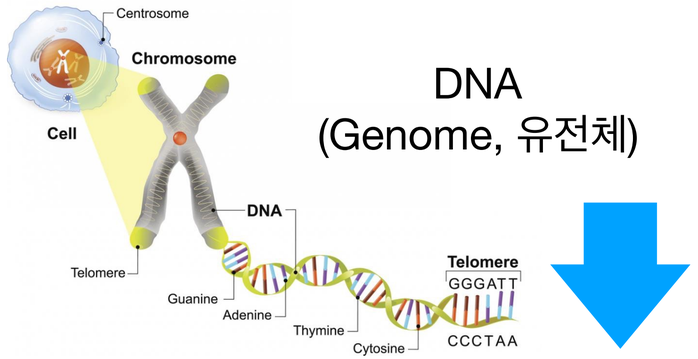
차례

소개

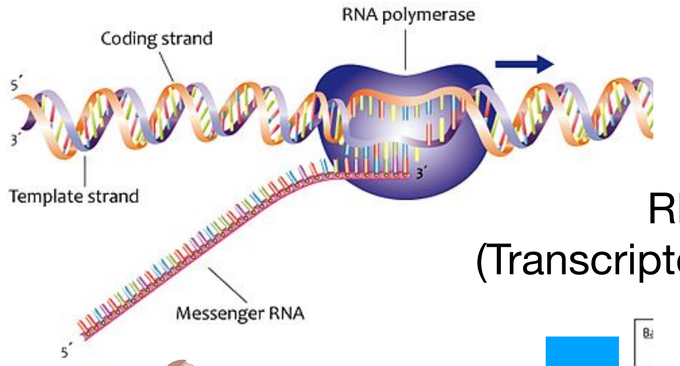
- At the heart of every living organism is its genome: the molecules of DNA containing all the instructions to make the organism's working parts.
- DNA is not just an abstract storage medium. It is a physical molecule that behaves in complicated ways. It also interacts with thousands of other molecules, all of which play important roles in maintaining, copying, directing, and carrying out the instructions contained in the DNA.

DNA, RNA, and Proteins

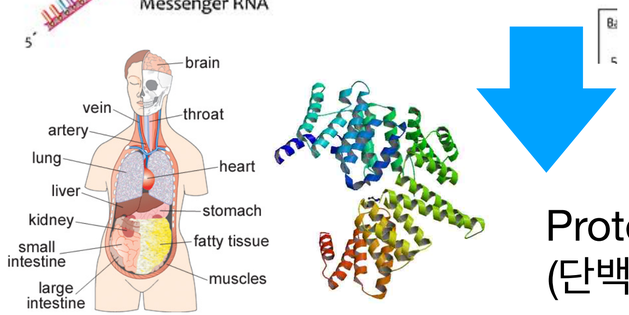
- DNA :
  - a long chain of repeating units strung together.
  - there are four units (called bases) that can appear: adenine, cytosine, guanine, and thymine, which are abbreviated as A, C, G, and T.
- RNA :
  - Going from DNA to protein involves another molecule that serves as an intermediate representation to carry information from one part of the cell to another.
  - RNA is yet another polymer and is chemically very similar to DNA.
  - It too has four bases that can be chained together in arbitrary orders.



DNA
(Genome, 유전체)



RNA
(Transcriptome, 전사체)



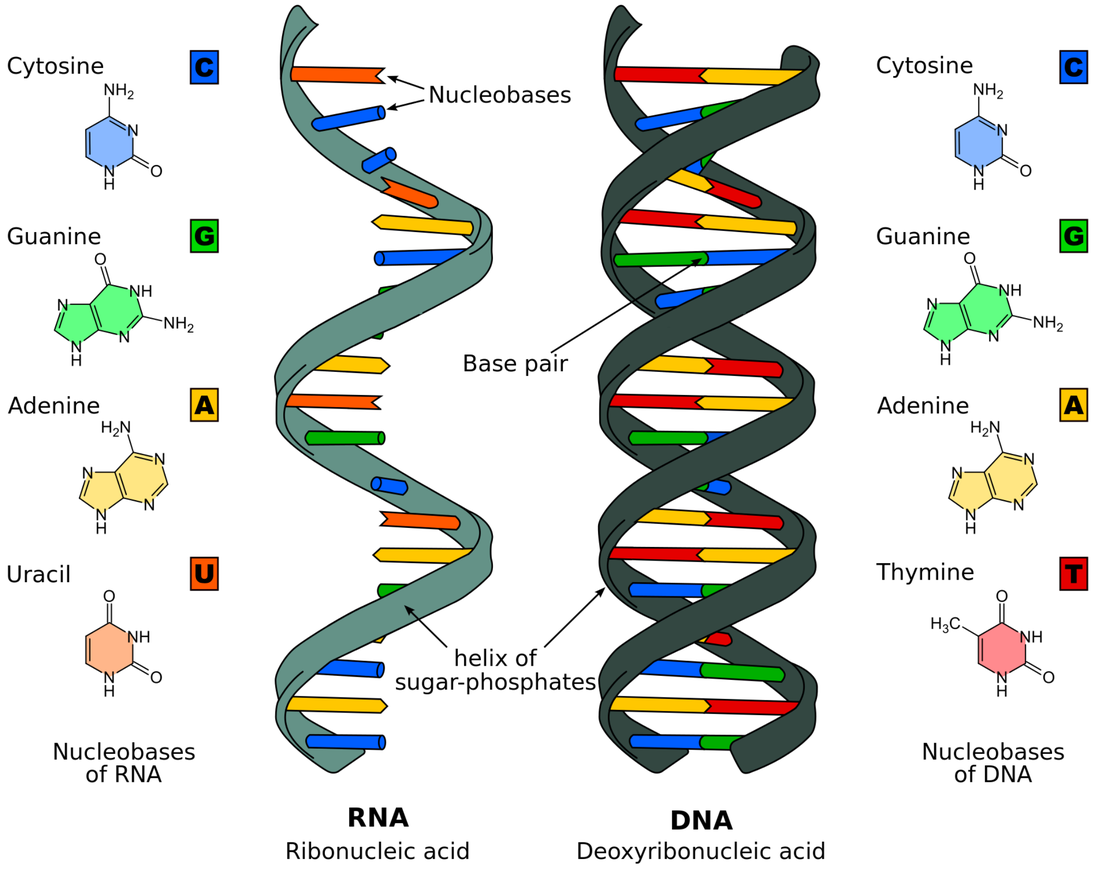
Protein
(단백질)

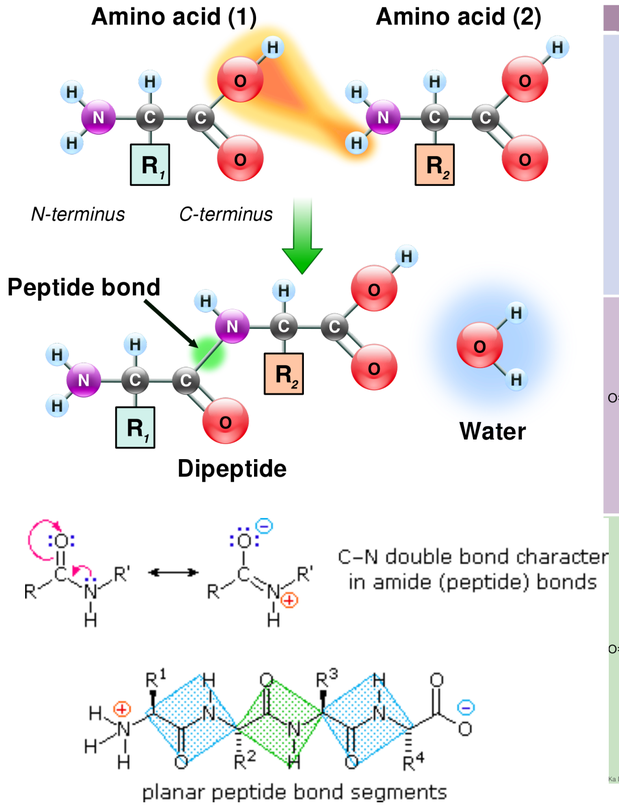
https://www.ldoceonline.com/dictionary/body

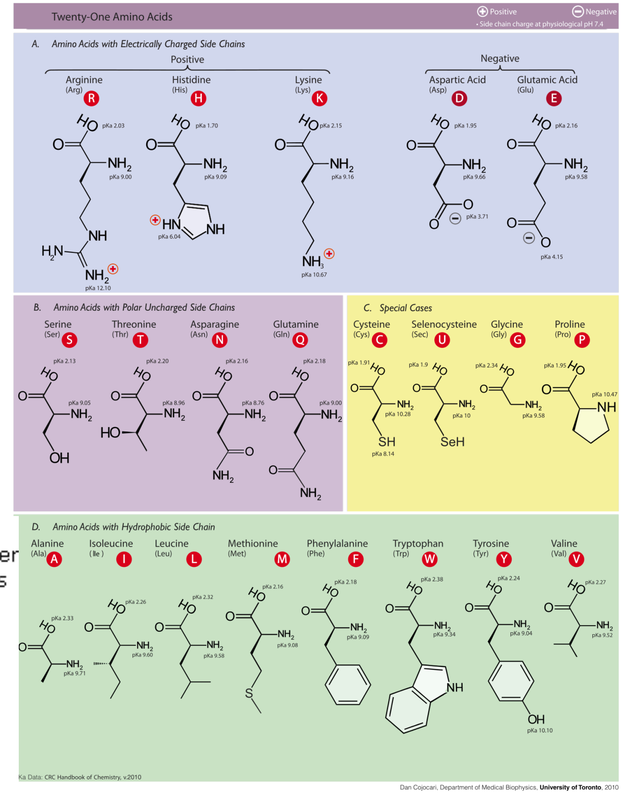
https://www.biology-questions-and-answers.com/protein-structure.html

ARONTIER

DNA, RNA, and Proteins







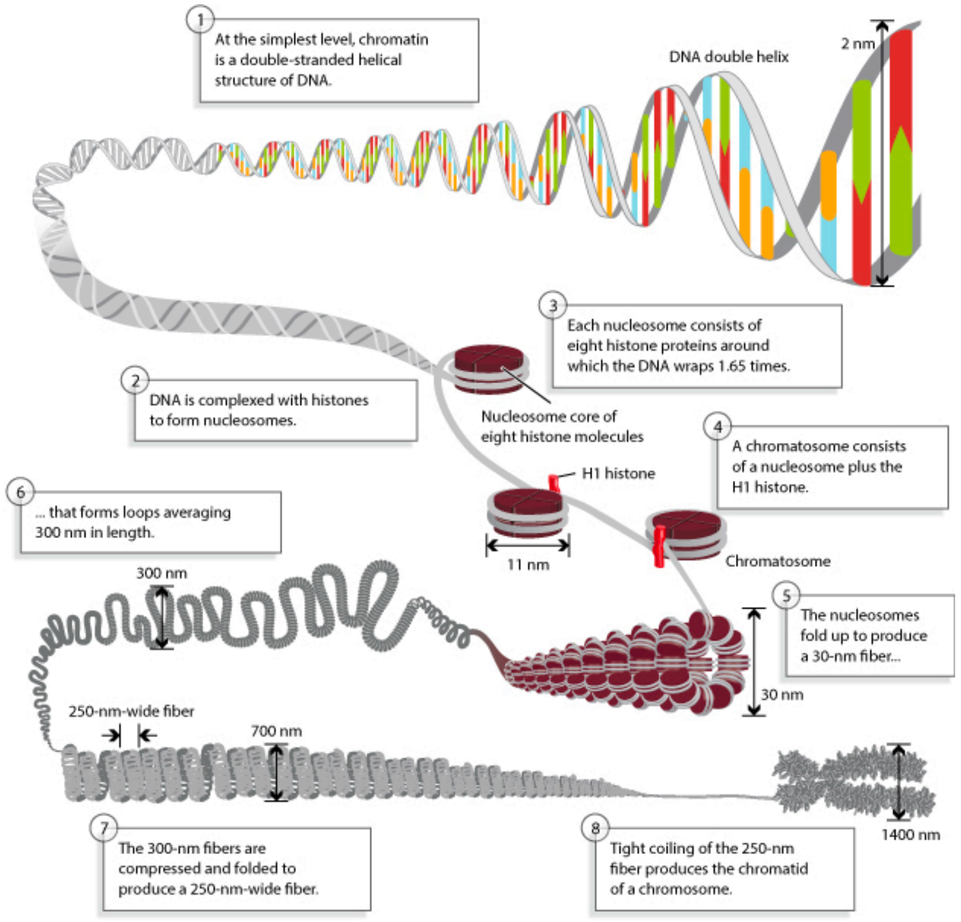
https://en.wikipedia.org/wiki/Peptide_bond

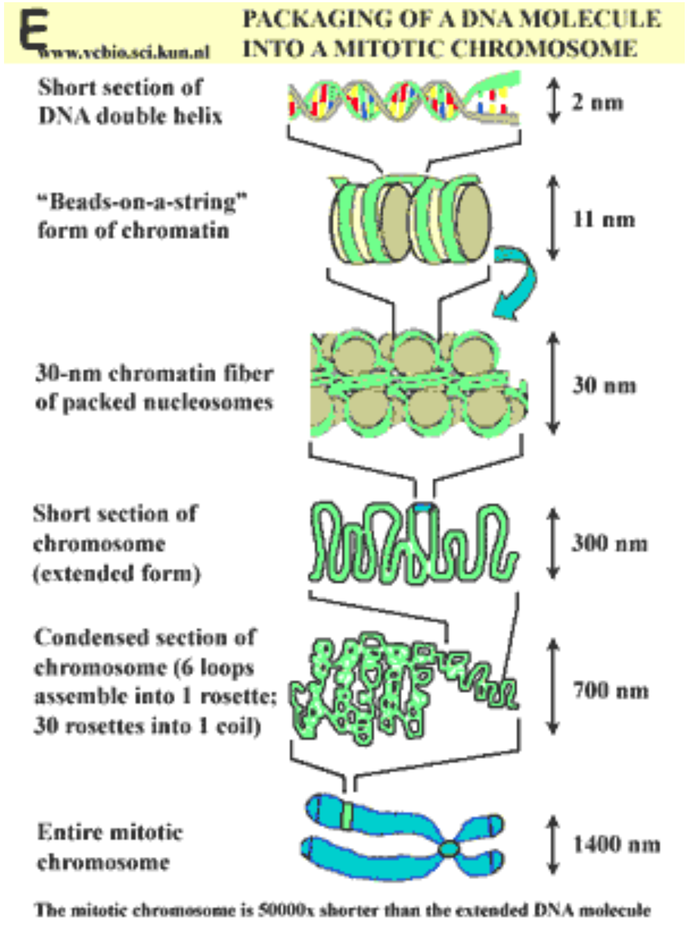
https://www2.chemistry.msu.edu/faculty/reusch/VirtTxtJml/protein2.htm

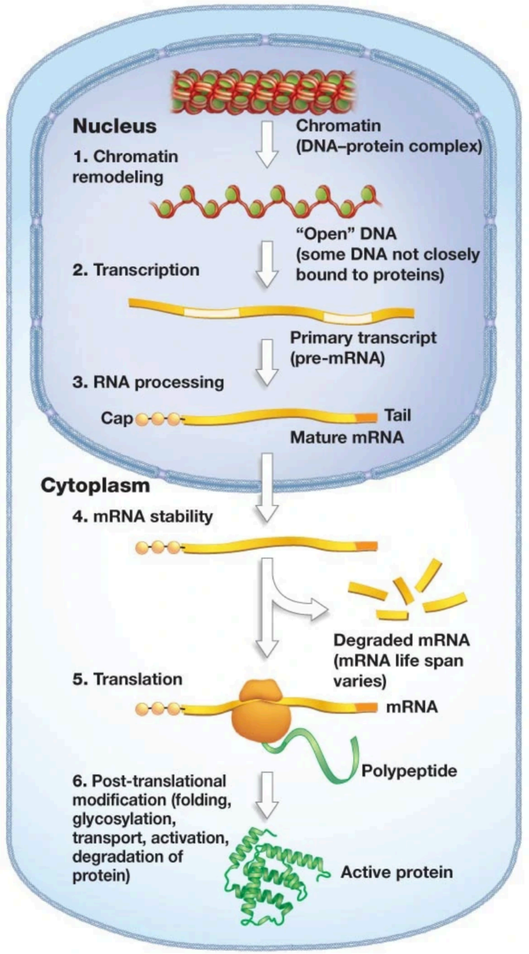
https://en.wikipedia.org/wiki/RNA_world#/media/File:Difference_DNA_RNA-EN.svg

ARONTIER

And Now for Real World







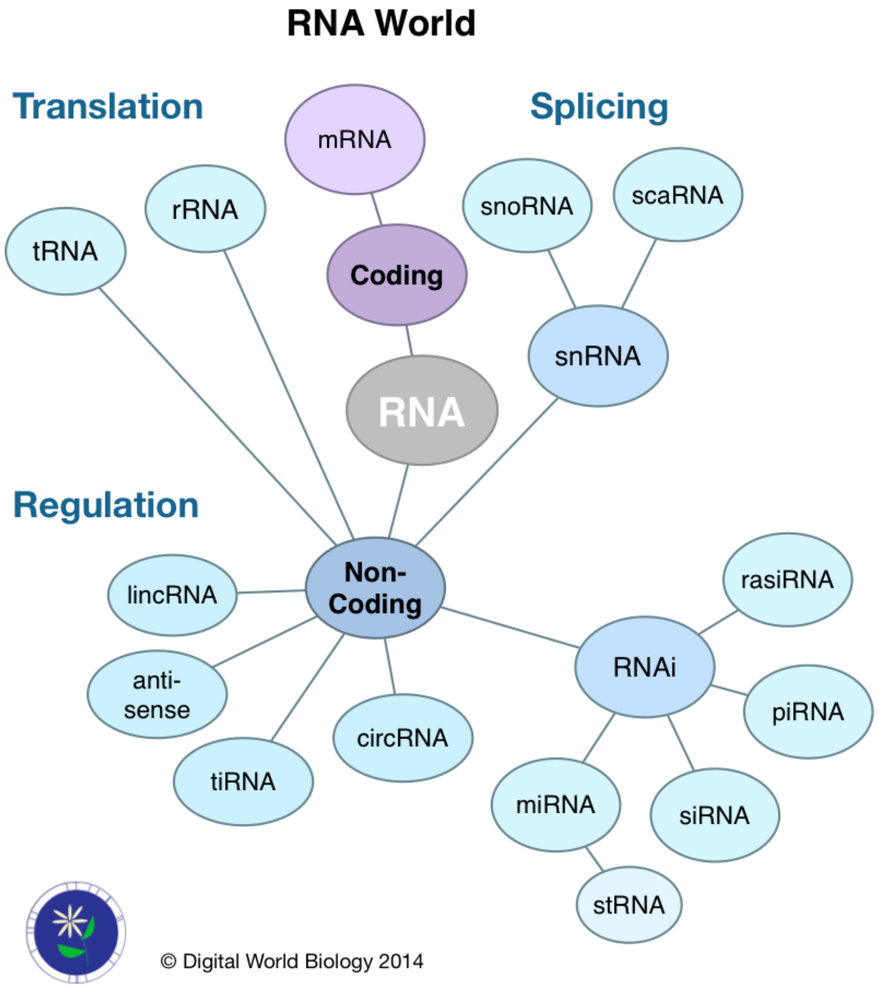
https://www.nature.com/scitable/topicpage/dna-packaging-nucleosomes-and-chromatin-310/

https://www.vcbio.science.ru.nl/en/virtuallessons/cellcycle/chromosome/

ARONTIER

RNA world

- Micro RNAs (miRNAs) are short pieces of RNA that bind to a messenger RNA and prevent it from being translated into proteins. This is a very important regulatory mechanism in some types of animals, especially mammals.
- Short interfering RNA (siRNA) is another type of RNA that binds to mRNA and prevents it from being translated. It's similar to miRNA, but siRNAs are double stranded (unlike miRNAs, which are single stranded), and some of the details of how they function are different. We will discuss siRNA in more detail later in the chapter.
- Ribozymes are RNA molecules that can act as enzymes to catalyze chemical reactions. Chemistry is the foundation of everything that happens in a living cell, so catalysts are vital to life. Usually this job is done by proteins, but we now know it sometimes is done by RNA.
- Riboswitches are RNA molecules that consist of two parts. One part acts as a messenger RNA, while the other part is capable of binding to a small molecule. When it binds, that can either enable or prevent translation of the mRNA. This is yet another regulatory mechanism by which protein production can be adjusted based on the concentration of particular small molecules in the cell.

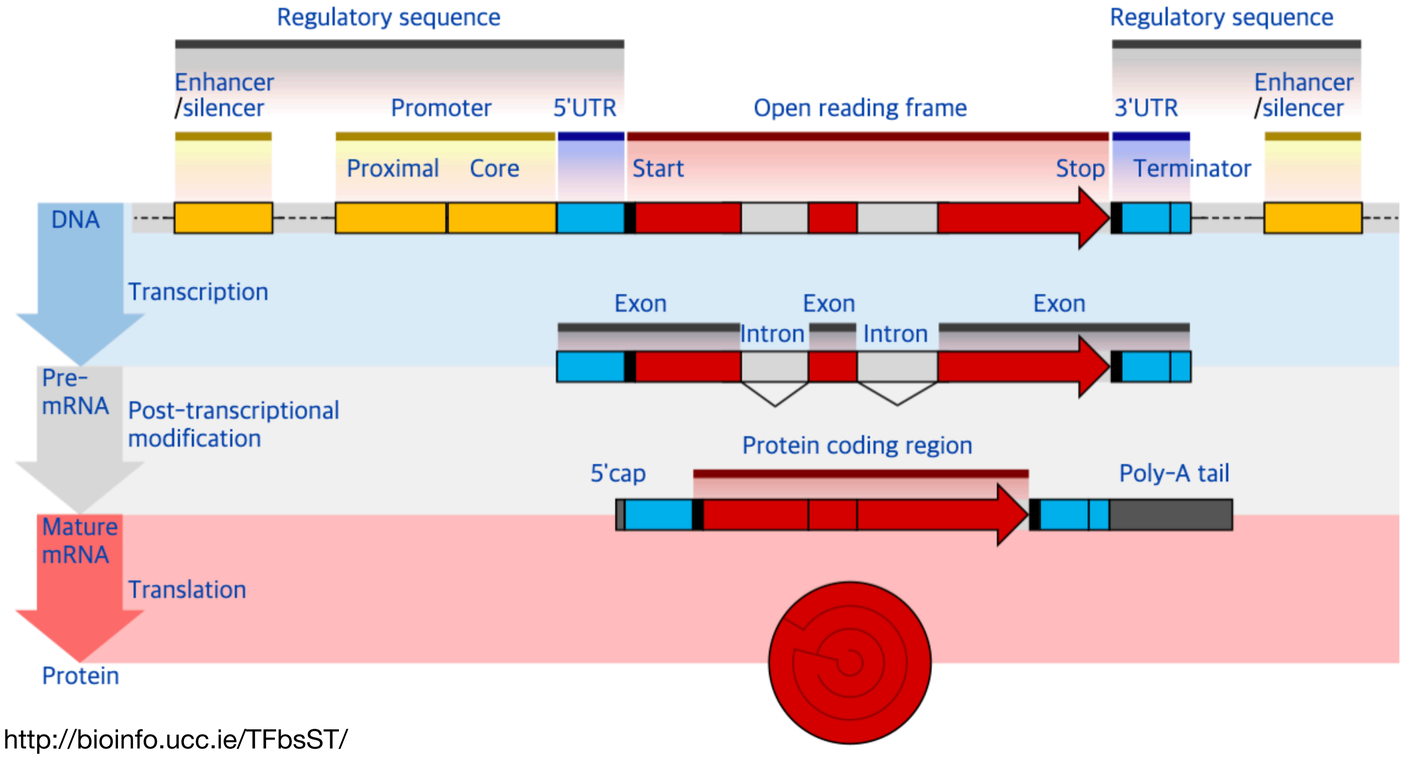


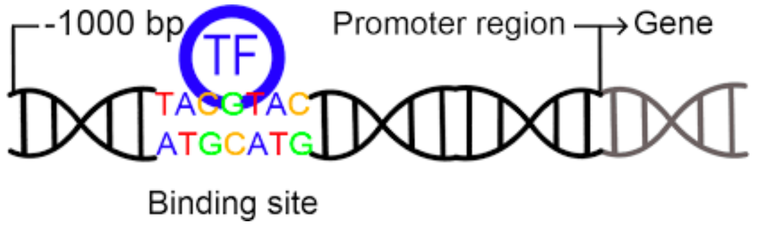
https://digitalworldbiology.com/archive/bio-databases-2014

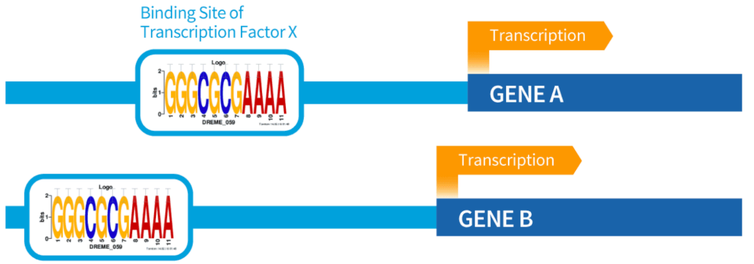
ARONTIER

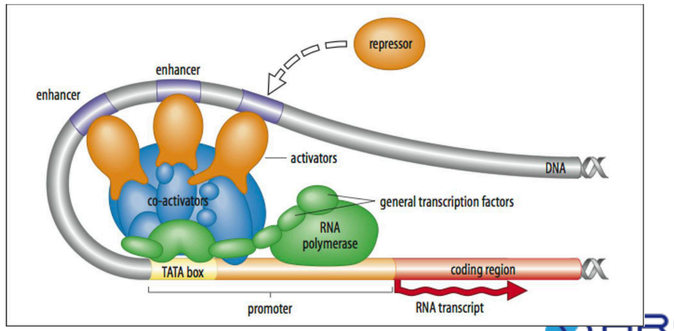
Transcription Factor Binding

- let's consider the problem of predicting transcription factor binding
- TFs are proteins that bind to DNA. When they bind, they influence the probability of nearby genes being transcribed into RNA









http://bioinfo.ucc.ie/TFbsST/

https://cage-seq.com/cage-analysis/index.html

https://en.wikipedia.org/wiki/Template:Eukaryote_gene_structure

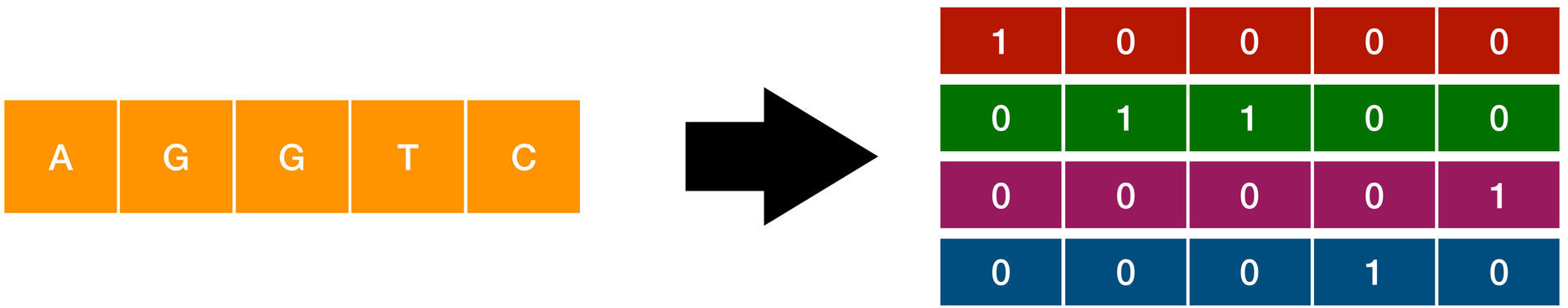
http://www.mun.ca/biology/desmid/brian/BIOL3530/DB_08/DBNDiff.html

A Convolutional Model for TF Binding

- We will use experimental data on a particular transcription factor called JUND.
  - An experiment was done to identify every place in the human genome where it binds.
  - To keep things manageable, we only include the data from chromosome 22, one of the smallest human chromosomes. It is still over 50 million bases long, so that gives us a reasonable amount of data to work with.
- The full chromosome has been split up into short segments, each 101 bases long, and each segment has been labeled to indicate whether it does or does not include a site where JUND binds.
  - We will try to train a model that predicts those labels based on the sequence of each segment.
  - The sequences are represented with one-hot encoding. For each base we have four numbers, of which one is set to 1 and the others are set to 0. Which of the four numbers is set to 1 indicates whether the base is an A, C, G, or T.

A Convolutional Model for TF Binding

- To process the data we will use a convolutional neural network, just like we did for recognizing handwritten digits in Chapter 3. In fact, you will see the two models are remarkably similar to each other.
- This time we will use 1D convolutions, since we are dealing with 1D data (DNA sequences) instead of 2D data (images), but the basic components of the model will be the same: inputs, a series of convolutional layers, one or more dense layers to compute the output, and a cross entropy loss function.



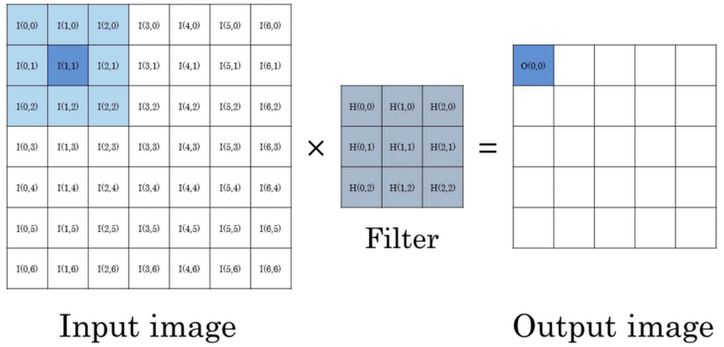
Creating TensorGraph and defining inputs

```
model = dc.models.TensorGraph(batch_size=1000)
features = layers.Feature(shape=(None, 101, 4))
labels = layers.Label(shape=(None, 1))
weights = layers.Weights(shape=(None, 1))
```

- a feature vector of size 101 (the number of bases) by 4 (the one-hot encoding of each base).
- We also have a single number for the label (either 0 or 1, to indicate whether it contains a binding site) and a single number for the weight.
- Using weights in the loss function is critical for this example,
  - because the data is very unbalanced. Less than 1% of all samples include a binding site.
  - That means the model could trivially get better than 99% accuracy by just outputting 0 for every sample. We prevent this by giving the positive samples higher weights than the negative ones.

Creating a stack of three convolutional layers

```
prev = features
for i in range(3):
    prev = layers.Conv1D(filters=15, kernel_size=10,
                         activation=tf.nn.relu, padding='same', in_layers=prev)
    prev = layers.Dropout(dropout_prob=0.5, in_layers=prev)
```



- The first layer takes the raw features (four numbers per base) as input.
- The second layer again looks at spans of 10 bases, but this time the inputs are the 15 values computed by the first layer. It computes a new set of 15 values for each base, and so on.
  - filters: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
- To prevent overfitting, we add a dropout layer after each convolutional layer. The dropout probability is set to 0.5, meaning that 50% of all output values are randomly set to 0.

```
keras.layers.Conv1D(filters, kernel_size, strides=1, padding='valid', data_format='channels_last', dilation_rate=1, activation=None,
use_bias=True, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None,
activity_regularizer=None, kernel_constraint=None, bias_constraint=None)
```

https://www.researchgate.net/figure/Image-convolution-with-an-input-image-of-size-7-7-and-a-filter-kernel-of-size-3-3_fig1_318849314

Using a dense layer to compute the output

```
logits = layers.Dense(out_channels=1, in_layers=layers.Flatten(prev))
output = layers.Sigmoid(logits)
model.add_output(output)
```

- We want the output to be between 0 and 1 so we can interpret it as the probability a particular sample contains a binding site.
- The dense layer can produce arbitrary values, not limited to any particular range. We therefore pass it through a logistic sigmoid function to compress it to the desired range.
- The input to this function is often referred to as logits.
  - The name refers to the mathematical logit function, which is the inverse function of the logistic sigmoid.

Computing the cross entropy for each sample and multiply by the weights to get the loss

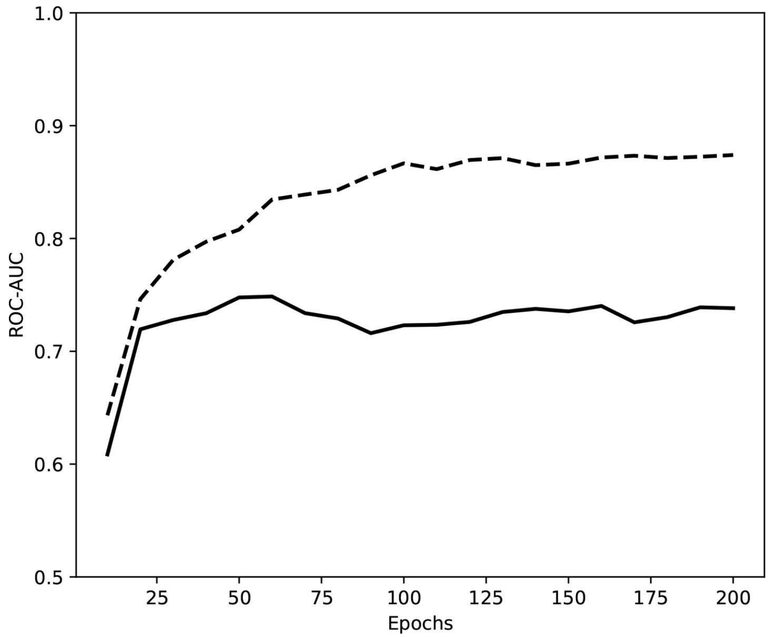
```
loss = layers.SigmoidCrossEntropy(in_layers=[labels, logits])
weighted_loss = layers.WeightedError(in_layers=[loss, weights])
model.set_loss(weighted_loss)
```

- Notice that for reasons of numerical stability, the cross entropy layer takes logits as input instead of the output of the sigmoid function.

Ready to train and evaluate the model

```
train = dc.data.DiskDataset('train_dataset')
valid = dc.data.DiskDataset('valid_dataset')
metric = dc.metrics.Metric(dc.metrics.roc_auc_score) for i in range(20):
    model.fit(train, nb_epoch=10)
    print(model.evaluate(train, [metric]))
print(model.evaluate(valid, [metric]))
```

- We use ROC AUC as our evaluation metric. After every 10 epochs of training, we evaluate the model on both the training and validation sets:
- The validation set performance peaks at about 0.75 after 50 epochs, then decreases slightly.
    - The training set performance continues to increase, eventually leveling off at around 0.87. This tells us that training beyond 50 epochs just leads to overfitting, and we should halt training at that point:



ARONTIER

Ready to train and evaluate the model

- A ROC AUC score of 0.75 is not bad, but also not wonderful.
- Possibly we could increase it by improving the model.
  - There are lots of hyperparameters we could try changing: the number of convolutional layers, the kernel width for each layer, the number of filters in each layer, the dropout rate, etc. We could try lots of combinations for them, and we might find one with better performance.
- But we also know there are fundamental limits to how well this model can ever work.
  - The only input it looks at is the DNA sequence, and TF binding also depends on lots of other factors: accessibility, methylation, shape, the presence of other molecules, etc. Any model that ignores those factors will be limited in how accurate its predictions can ever be.
- So now let's try adding a second input and see if it helps.

Chromatin Accessibility

- Chromatin refers to everything that makes up a chromosome:
  - DNA, histones, and various other proteins and RNA molecules.
- Chromatin accessibility refers to how accessible each part of the chromosome is to outside molecules.
  - When the DNA is tightly wound around histones, is becomes inaccessible to transcription factors and other molecules. They cannot reach it, and the DNA is effectively inactive.
  - When it unwinds from the histones, it becomes accessible again and resumes its role as a central part of the cell's machinery.
- Chromatin accessibility is neither uniform nor static. It varies between types of cells and stages of a cell's life cycle.
  - It can be affected by environmental conditions.
  - Instead of thinking of accessibility as a binary choice (accessible or inaccessible), it is better to think of it as a continuous variable (what fraction of the time each region is accessible).

Chromatin Accessibility

- Accessibility also is constantly changing as DNA winds and unwinds in response to events in the cell.
  - Instead of thinking of accessibility as a binary choice (accessible or inaccessible), it is better to think of it as a continuous variable (what fraction of the time each region is accessible).
- The data we analyzed in the last section came from experiments on a particular kind of cell called HepG2.
  - The experiments identified locations in the genome where the transcription factor JUND was bound.
- The results were influenced by chromatin accessibility. If a particular region is almost always inaccessible in HepG2 cells, the experiment was very unlikely to find JUND bound there, even if the DNA sequence would otherwise be a perfect binding site. So, let's try incorporating accessibility into our model.

Chromatin Accessibility

```
span_accessibility = {}
for line in open('accessibility.txt'):
  fields = line.split()
  span_accessibility[fields[0]] = float(fields[1])
```

- First let's load some data on accessibility.
- We have it in a text file where each line corresponds to one sample from our dataset (a 101-base stretch of chromosome 22). A line contains the sample ID followed by a number that measures how accessible that region tends to be in HepG2 cells. We'll load it into a Python dictionary:

To build the model

```
logits = layers.Dense(out_channels=1,in_layers=layers.Flatten(prev))

accessibility = layers.Feature(shape=(None, 1))

prev = layers.Concat([layers.Flatten(prev), accessibility])
logits = layers.Dense(out_channels=1, in_layers=prev)
```

- This time we will do the same thing, but also append the accessibility to the output of the convolution

Time to train it

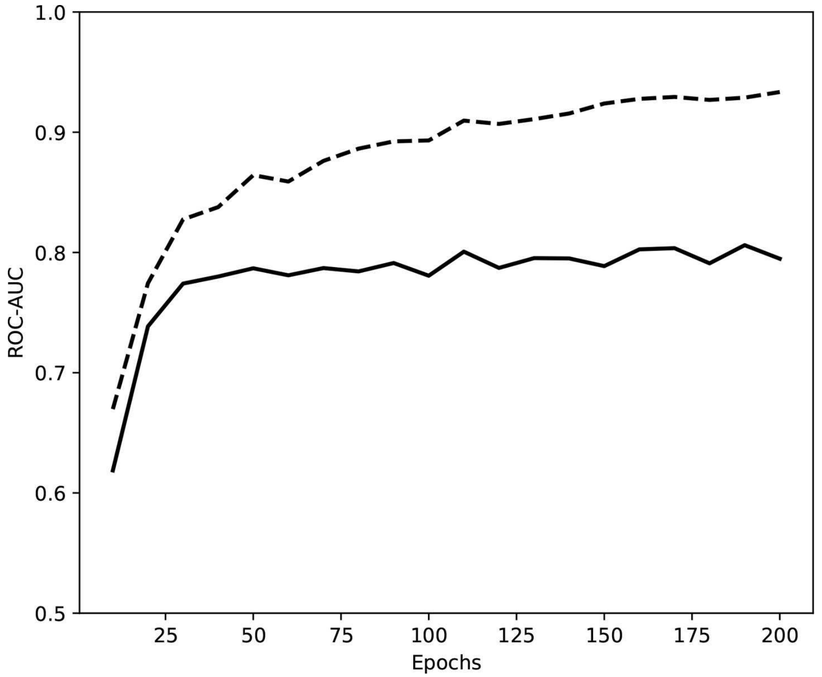
```
def generate_batches(dataset, epochs):
    for epoch in range(epochs):
        for X, y, w, ids in dataset.iterbatches(batch_size=1000, pad_batches=True):
            yield {
                features: X,
                accessibility: np.array([span_accessibility[id] for id in ids]),
                labels: y,
                weights: w
            }
```

- At this point we run into a difficulty:
  - our model has two different Feature layers!

Time to train it

```
for i in range(20):
    model.fit_generator(generate_batches(train, 10))
    print(model.evaluate_generator(generate_batches(train, 1),[metric],labels=[labels], weights=[weights]))
    print(model.evaluate_generator(generate_batches(valid, 1),[metric],labels=[labels], weights= [weights]))
```

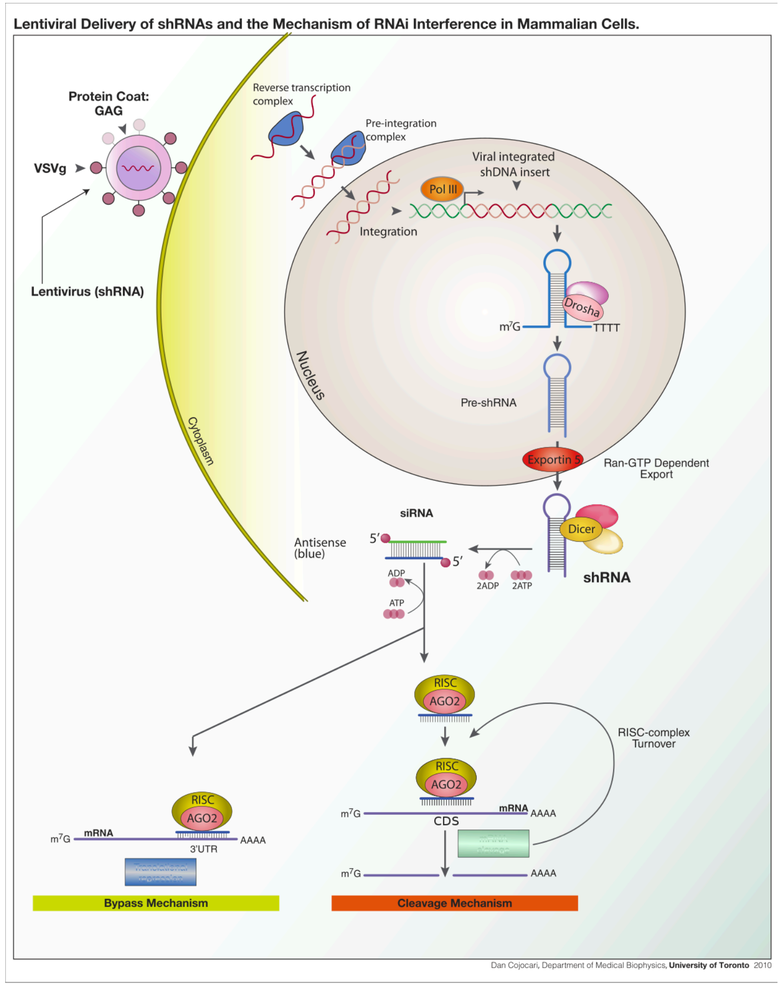
- Both the training and validation set scores are improved compared to the model that ignored chromatin accessibility.
- ROC AUC score now reaches 0.91 for the training set and 0.80 for the validation set.



ARONTIER

RNA Interference

- RNA interference.
  - A short piece of RNA whose sequence is complementary to part of a messenger RNA can bind to that mRNA.
  - When this happens, it "silences" the mRNA and prevents it from being translated into a protein.
  - The molecule that does the silencing is called a short interfering RNA (siRNA).
- RNA interference is a complex biological mechanism
  - It begins with the siRNA binding to a collection of proteins called the RNA-induced silencing complex (RISC).
  - The RISC uses the siRNA as a template to search out matching mRNAs in the cell and degrade them.



https://ko.wikipedia.org/wiki/RNA_%EA%B0%84%EC%84%AD

ARONTIER

Dataset

- We'll train our model using a library of 2,431 siRNA molecules, each 21 bases long.
- Every one of them has been tested experimentally and labeled with a value between 0 and 1, indicating how effective it is at silencing its target gene.
- Small values indicate ineffective molecules, while larger values indicate more effective ones. The model takes the sequence as input and tries to predict the effectiveness.

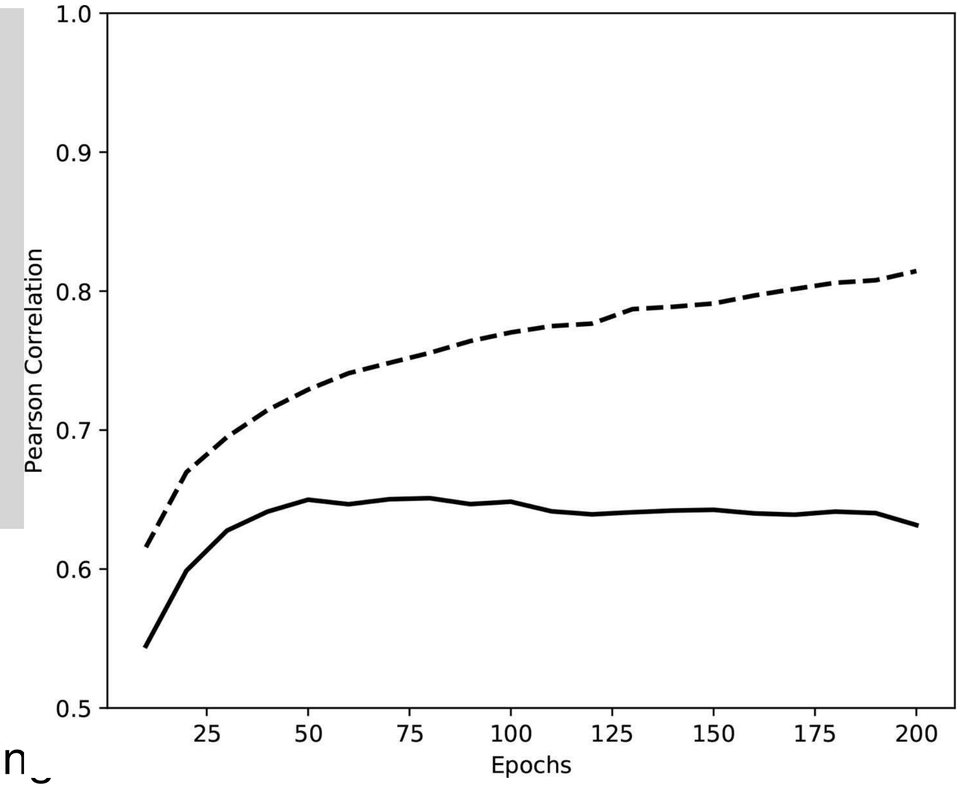
To build the model

```
model = dc.models.TensorGraph()
features = layers.Feature(shape=(None, 21, 4))
labels = layers.Label(shape=(None, 1)) prev = features
for i in range(2):
    prev = layers.Conv1D(filters=10, kernel_size=10, activation=tf.nn.relu, padding='same', in_layers=prev)
    prev = layers.Dropout(dropout_prob=0.3, in_layers=prev)
output = layers.Dense(out_channels=1, activation_fn=tf.sigmoid, in_layers=layers.Flatten(prev))
model.add_output(output)
loss = layers.ReduceMean(layers.L2Loss(in_layers=[labels,output]))
model.set_loss(loss)
```

- This is very similar to the model we used for TF binding, with just a few differences
- We also use a different loss function.
  - this one is a regression model.

Training

```
train = dc.data.DiskDataset('train_siRNA')
valid = dc.data.DiskDataset('valid_siRNA')
metric = dc.metrics.Metric(dc.metrics.pearsonr,mode='regression')
for i in range(20):
    model.fit(train, nb_epoch=10)
    print(model.evaluate(train, [metric]))
    print(model.evaluate(valid, [metric]))
```



- The validation set score peaks at 0.65 after 50 epochs.
- The training set score continues to increase, but since there is no further improvement to the validation set score this is just overfitting. Given the simplicity of the model and the limited amount of training data, a correlation coefficient of 0.65 is quite good.
- More complex models trained on larger datasets do slightly better, but this is already very respectable performance.

END