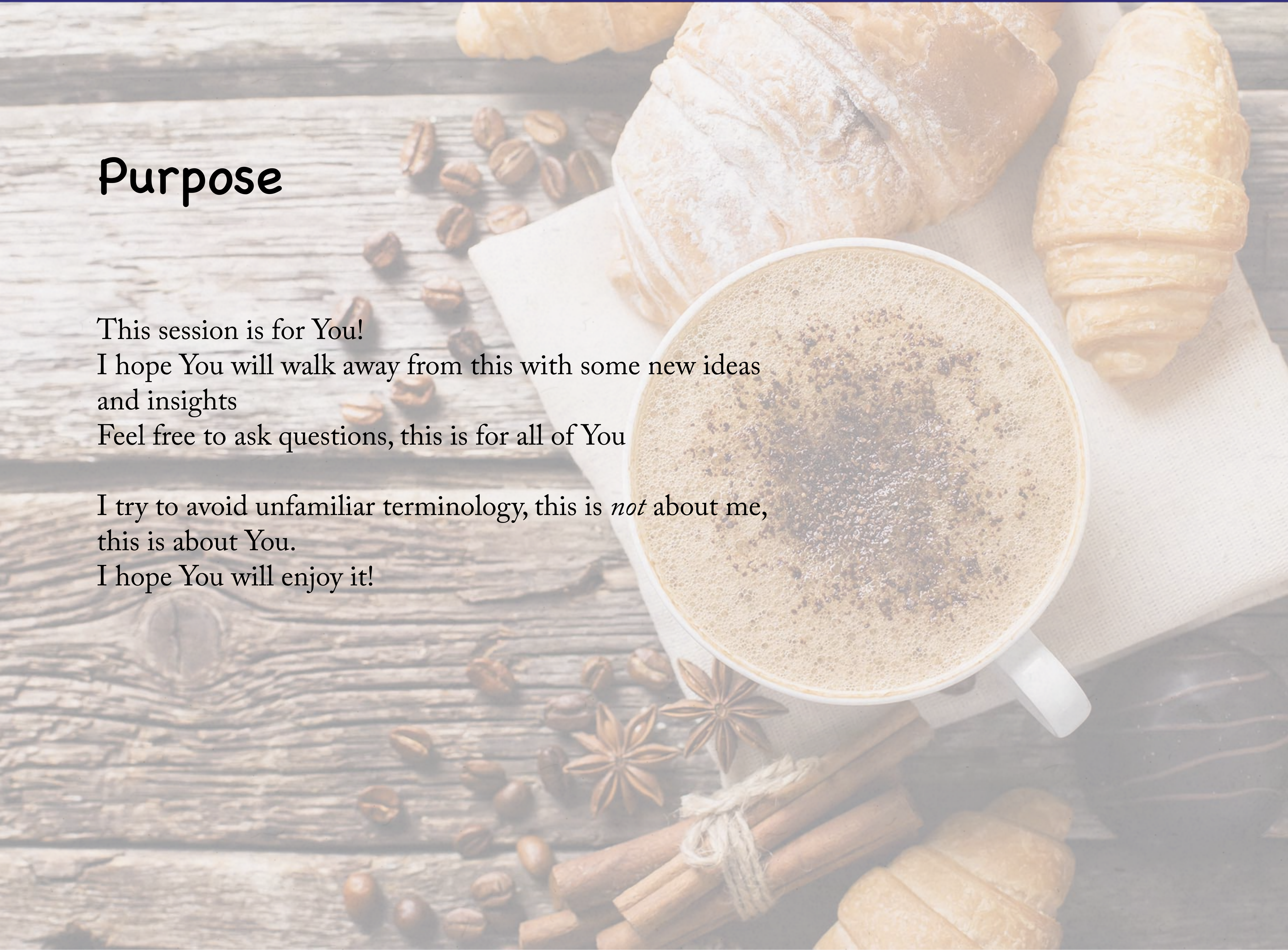
MACHINE LEARNING ON THE JAVA PLATFORM

DAVID STRÖM

2019-09-12 | CALLISTAENTERPRISE.SE

CALLISTA
— ENTERPRISE —



Purpose

This session is for You!
I hope You will walk away from this with some new ideas and insights
Feel free to ask questions, this is for all of You

I try to avoid unfamiliar terminology, this is *not* about me, this is about You.
I hope You will enjoy it!

AGENDA

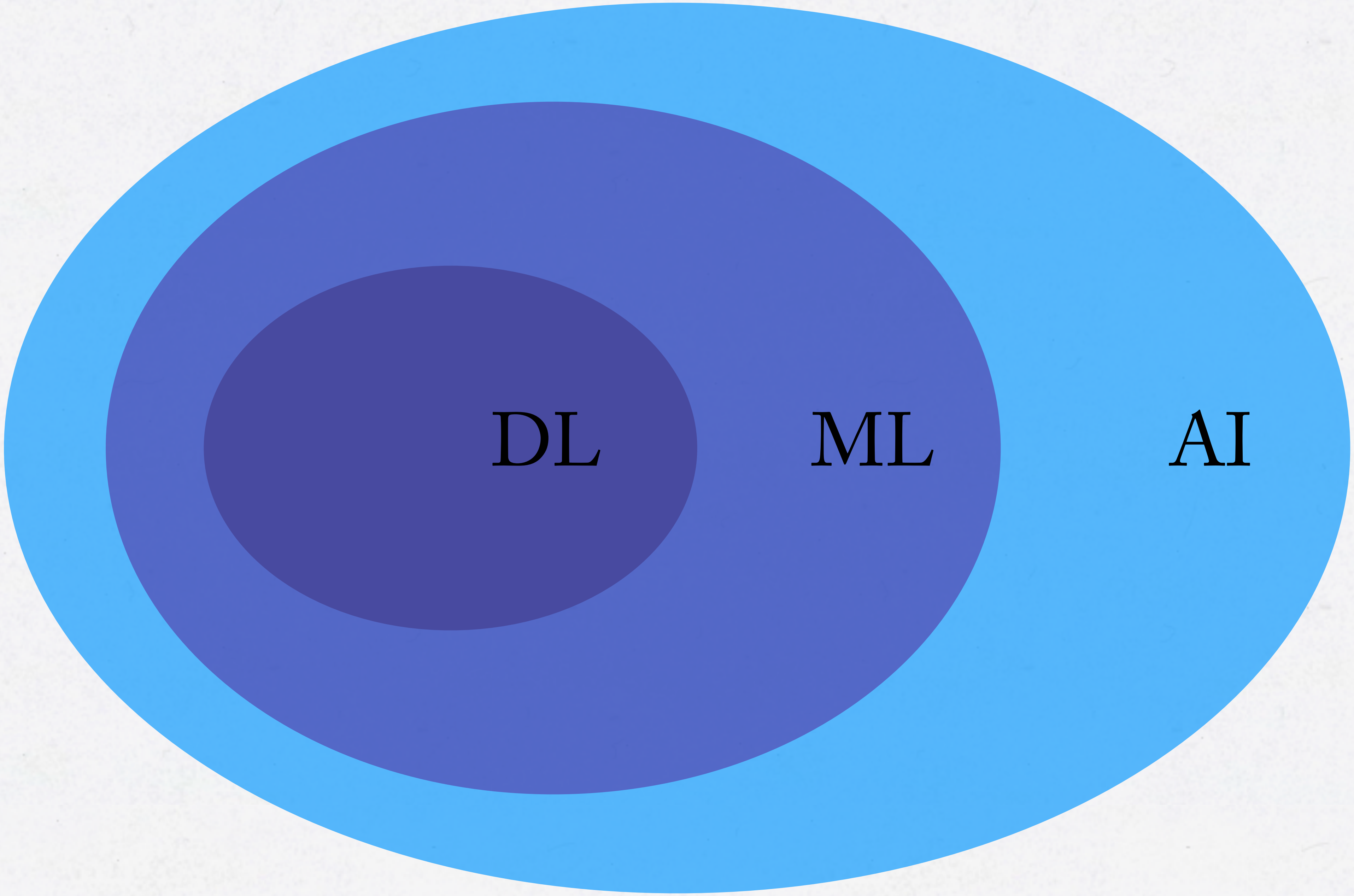
- The incredible power of machine learning
- Common machine learning challenges
- How Java faces up to those challenges
- How to implement a machine learning system in Java
- Looking ahead
- Summary

THE INCREDIBLE POWER OF MACHINE LEARNING

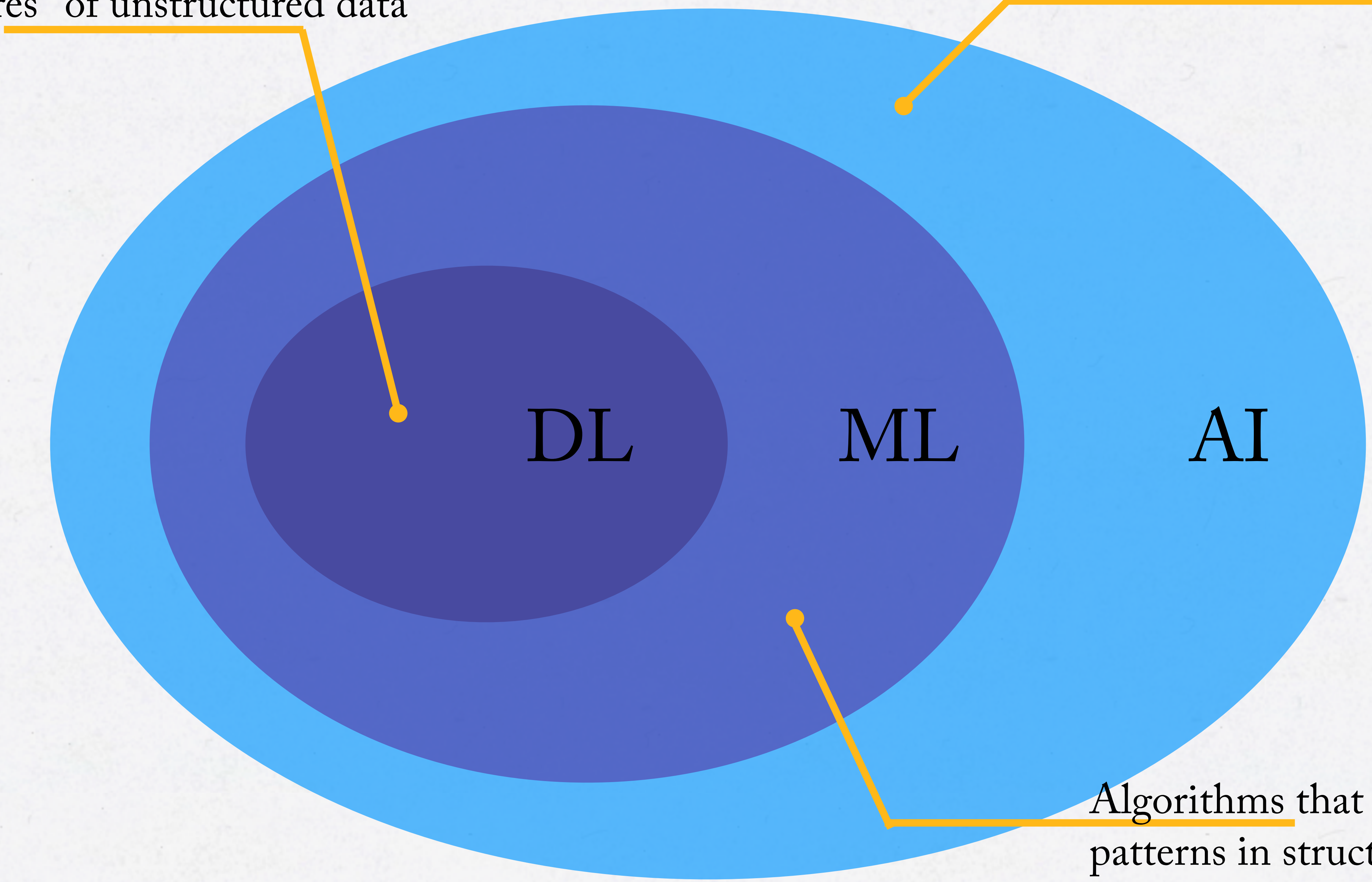
WHAT PROBLEMS CAN ML SOLVE?

- Some things are almost impossible to solve without ML, e.g.
  - Audio and image recognition
- While other things can get (a lot) better
  - Interpreting human language
  - Recommendations
  - Predictions
  - Anomaly detection
- Patterns are everywhere
  - What can your data tell you?

ARE ALL MACHINE LEARNING ALGORITHMS EQUAL?



ARE ALL MACHINE LEARNING ALGORITHMS EQUAL?



Algorithms that can learn "deep features" of unstructured data

Any algorithms that makes a system seem smart. No "learning"

Algorithms that can learn patterns in structured data

WHY JAVA?

- Date.from(Instant.now()): {Python} > {Java} < {R}
- Java is versatile with huge ecosystem of tools
- My thesis: As ML moves toward more and more practical implementation instead of research, a system development approach is needed

MACHINE LEARNING IN JAVA

- Deeplearning4J
- Weka
- Apache Mahout
- JavaML
- Etc.

MACHINE LEARNING IN JAVA

- Open source
- Includes various tools for ML
  - ND4J
  - DataVec
  - Arbiter
  - Some visualisation tools
- Import of Keras models
- Support dataprocessing on CUDA* enabled GPUs

* CUDA: Compute Unified Device Architecture

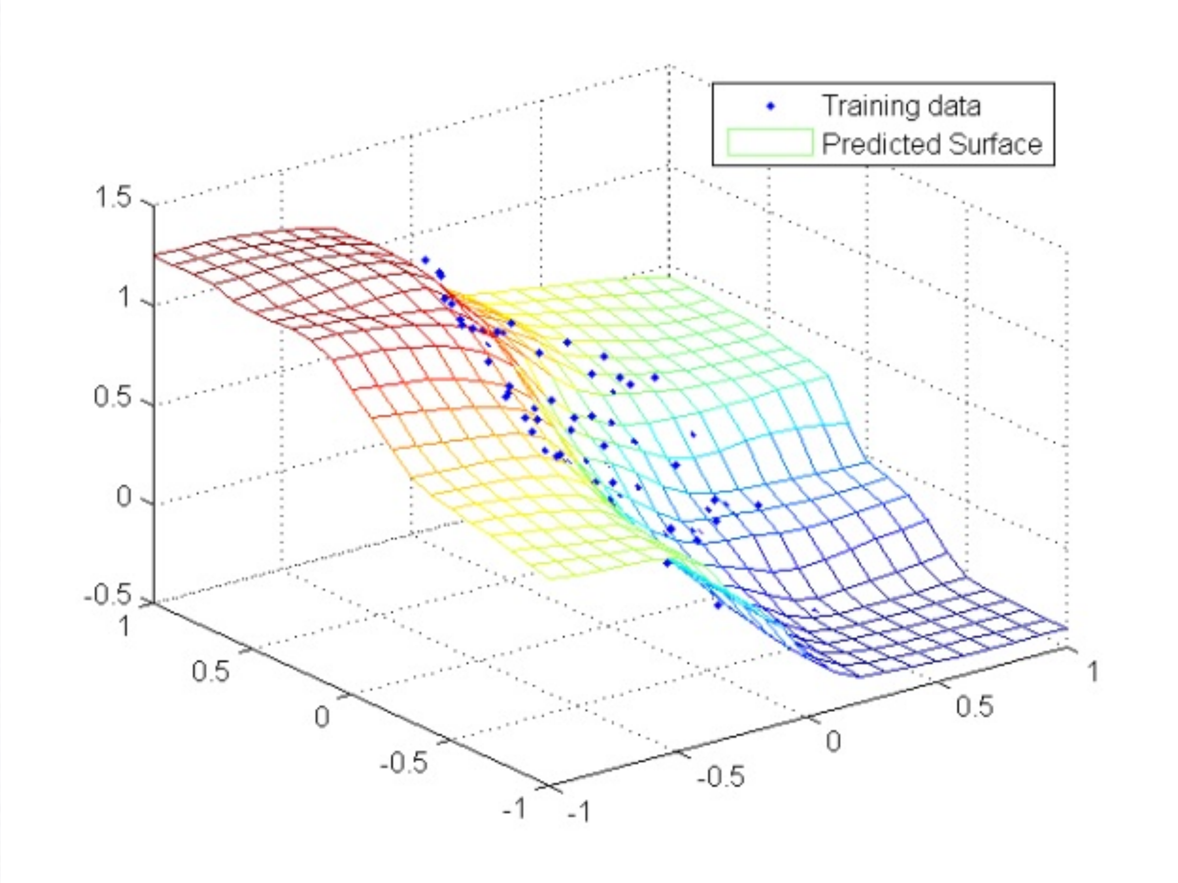
COMMON MACHINE LEARNING CHALLENGES

CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function…

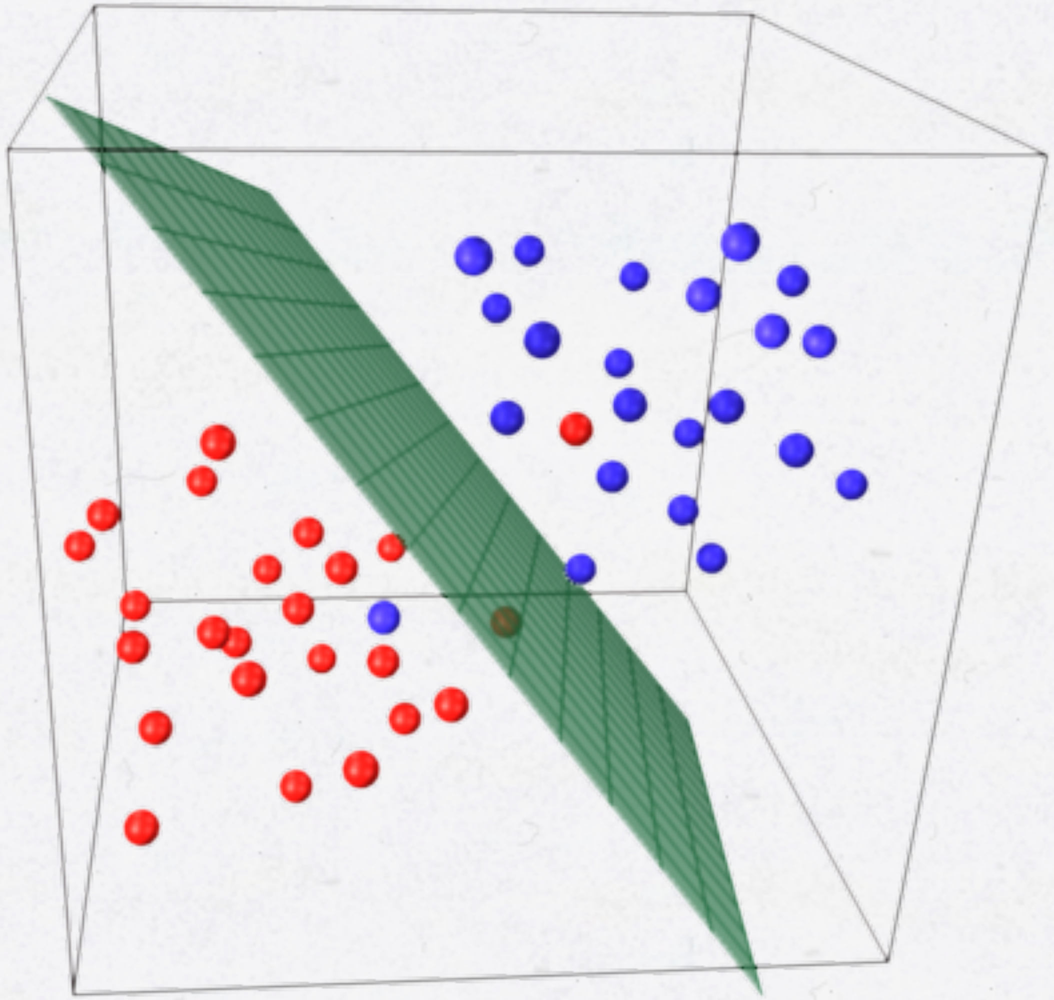
CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function...



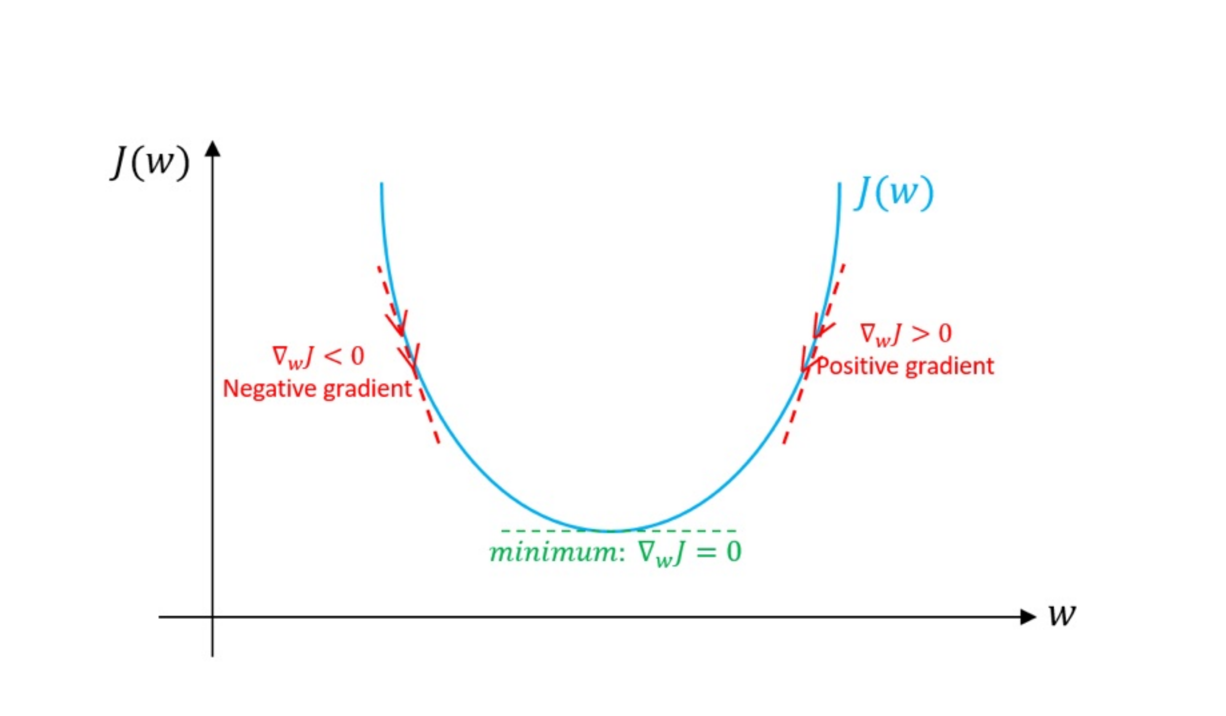
CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function...



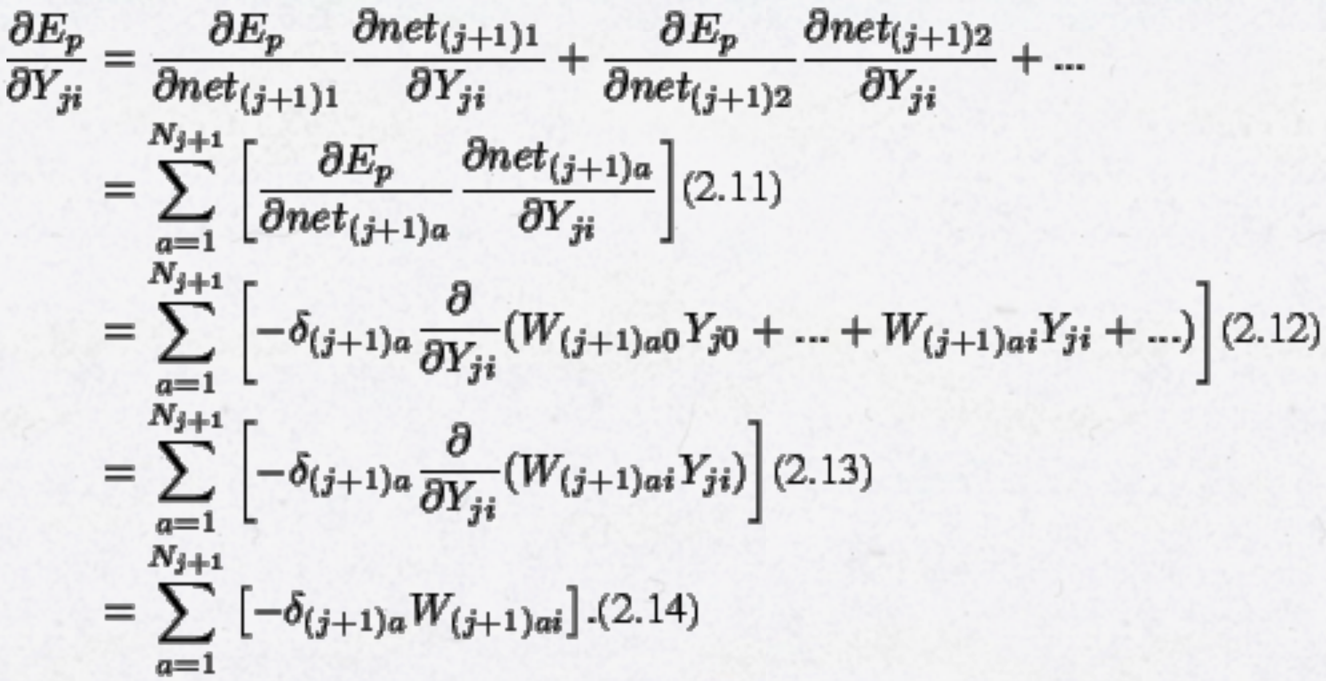
CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function…



CHALLENGE #1: COMPLEXITY

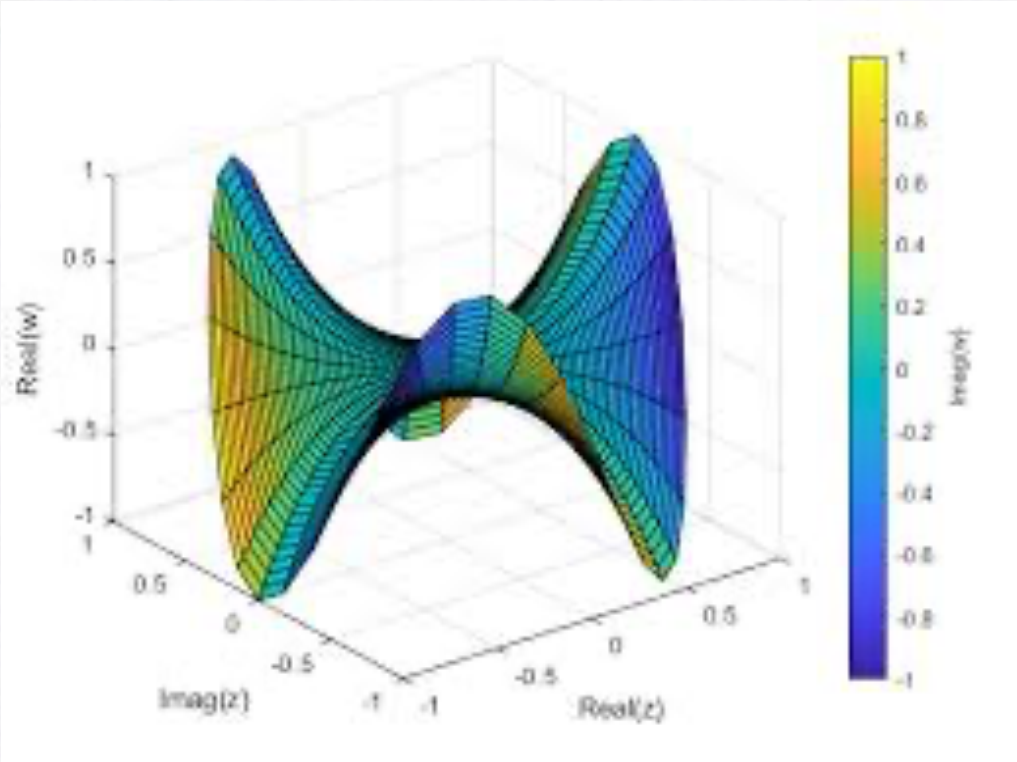
Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function...



$$
\begin{aligned}
\frac{\partial E_p}{\partial Y_{ji}} &= \frac{\partial E_p}{\partial net_{(j+1)1}} \frac{\partial net_{(j+1)1}}{\partial Y_{ji}} + \frac{\partial E_p}{\partial net_{(j+1)2}} \frac{\partial net_{(j+1)2}}{\partial Y_{ji}} + \dots \\
&= \sum_{a=1}^{N_{j+1}} \left[\frac{\partial E_p}{\partial net_{(j+1)a}} \frac{\partial net_{(j+1)a}}{\partial Y_{ji}} \right] (2.11) \\
&= \sum_{a=1}^{N_{j+1}} \left[-\delta_{(j+1)a} \frac{\partial}{\partial Y_{ji}} (W_{(j+1)a0} Y_{j0} + \dots + W_{(j+1)ai} Y_{ji} + \dots) \right] (2.12) \\
&= \sum_{a=1}^{N_{j+1}} \left[-\delta_{(j+1)a} \frac{\partial}{\partial Y_{ji}} (W_{(j+1)ai} Y_{ji}) \right] (2.13) \\
&= \sum_{a=1}^{N_{j+1}} \left[-\delta_{(j+1)a} W_{(j+1)ai} \right] . (2.14)
\end{aligned}
$$

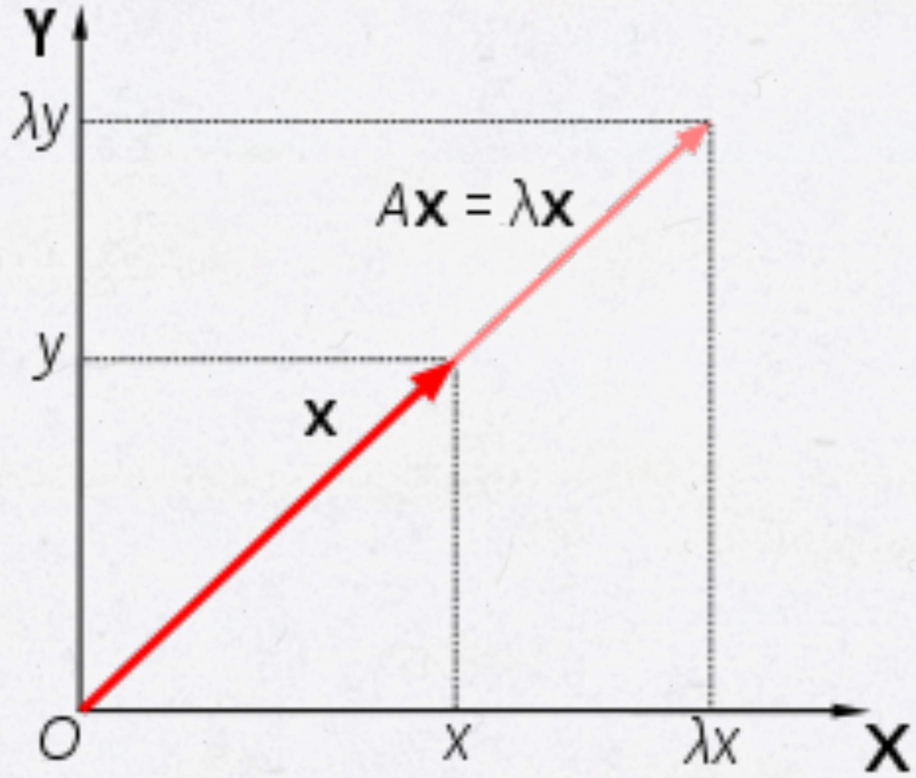
CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function…



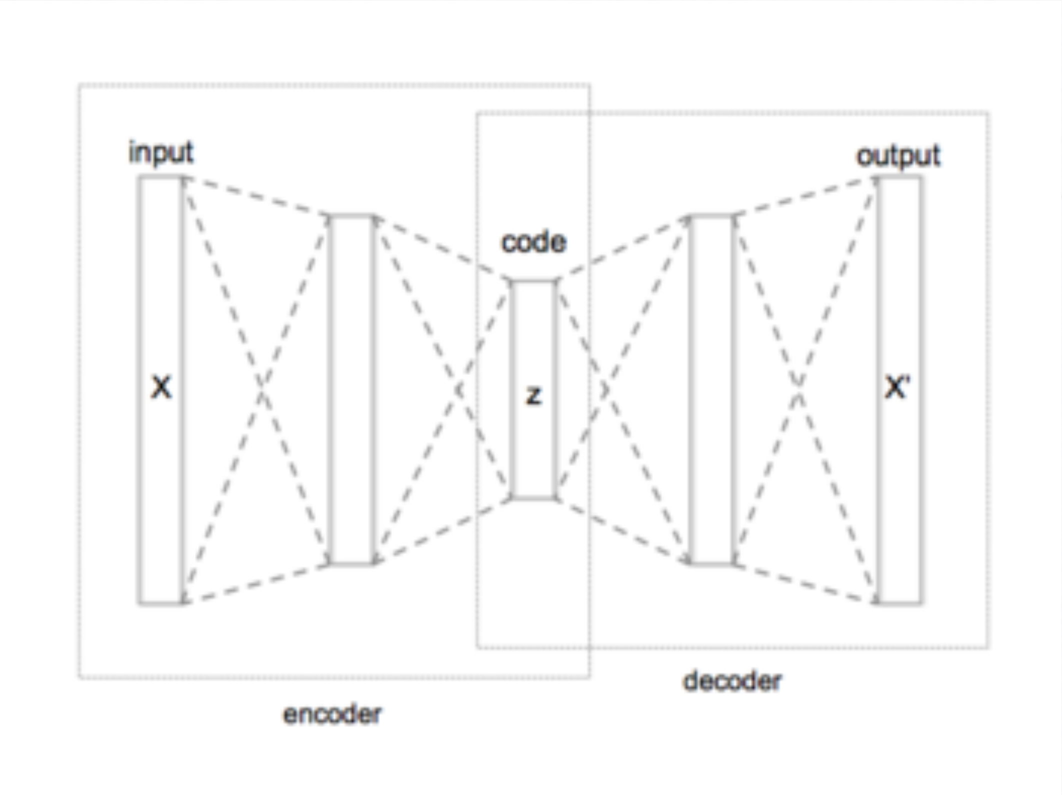
CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function…



CHALLENGE #1: COMPLEXITY

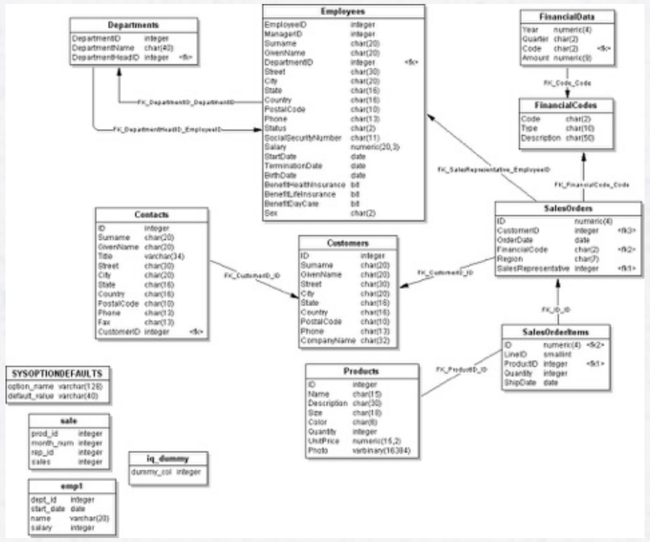
Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function...



CHALLENGE #1: COMPLEXITY

Backpropagation, naive Bayes, convolutional neural networks, decision trees, autoencoders, generative adversial networks, linear regression, eigenvectors, activation function, hyperparameters, classification, LSTM, random forest, support vector machines, restricted Boltzmann machine, interpolation, K- nearest, logistic regression, gradient descent, transpose, sequence padding, vectorization, polynomials, inference, hyperspace, n-dimensional arrays, learning rate, loss function…

CHALLENGE #2: DATA, DATA, DATA…



Departments		
DepartmentID	integer	
DepartmentName	char(40)	
DepartmentHeadID	integer	<fk>

Employees		
EmployeeID	integer	
ManagerID	integer	
Surname	char(20)	
GivenName	char(20)	
DepartmentID	integer	<fk>
Street	char(30)	
City	char(20)	
State	char(16)	
Country	char(16)	
PostalCode	char(10)	
Phone	char(13)	
Status	char(2)	
SocialSecurityNumber	char(11)	
Salary	numeric(20,3)	
StartDate	date	
TerminationDate	date	
BirthDate	date	
BenefitHealthInsurance	bit	
BenefitLifeInsurance	bit	
BenefitDayCare	bit	
Sex	char(2)	

FinancialData		
Year	numeric(4)	
Quarter	char(2)	
Code	char(2)	<fk>
Amount	numeric(9)	

FinancialCodes	
Code	char(2)
Type	char(10)
Description	char(50)

Contacts		
ID	integer	
Surname	char(20)	
GivenName	char(20)	
Title	varchar(34)	
Street	char(30)	
City	char(20)	
State	char(16)	
Country	char(16)	
PostalCode	char(10)	
Phone	char(13)	
Fax	char(13)	
CustomerID	integer	<fk>

Customers	
ID	integer
Surname	char(20)
GivenName	char(20)
Street	char(30)
City	char(20)
State	char(16)
Country	char(16)
PostalCode	char(10)
Phone	char(13)
CompanyName	char(32)

SalesOrders		
ID	numeric(4)	
CustomerID	integer	<fk3>
OrderDate	date	
FinancialCode	char(2)	<fk2>
Region	char(7)	
SalesRepresentative	integer	<fk1>

SalesOrderItems		
ID	numeric(4)	<fk2>
LineID	smallint	
ProductID	integer	<fk1>
Quantity	integer	
ShipDate	date	

Products	
ID	integer
Name	char(15)
Description	char(30)
Size	char(18)
Color	char(6)
Quantity	integer
UnitPrice	numeric(15,2)
Photo	varbinary(16384)

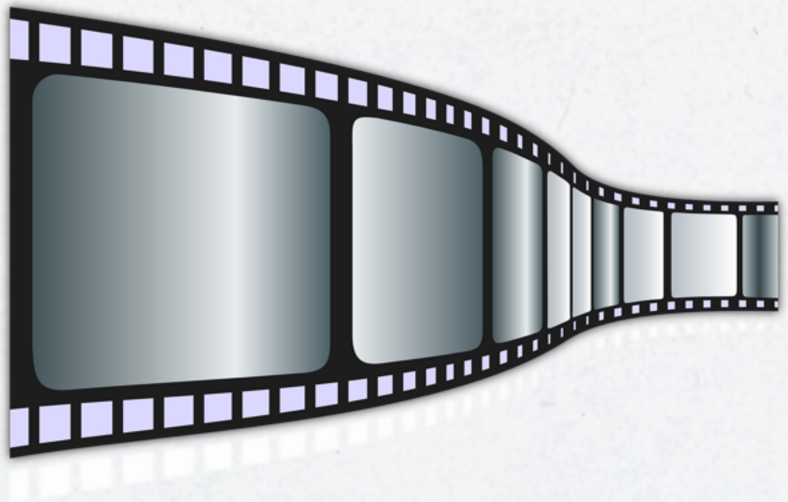
SYSOPTIONDEFAULTS	
option_name	varchar(128)
default_value	varchar(40)

sale	
prod_id	integer
month_num	integer
rep_id	integer
sales	integer

iq_dummy	
dummy_col	integer

emp1	
dept_id	integer
start_date	date
name	varchar(20)
salary	integer

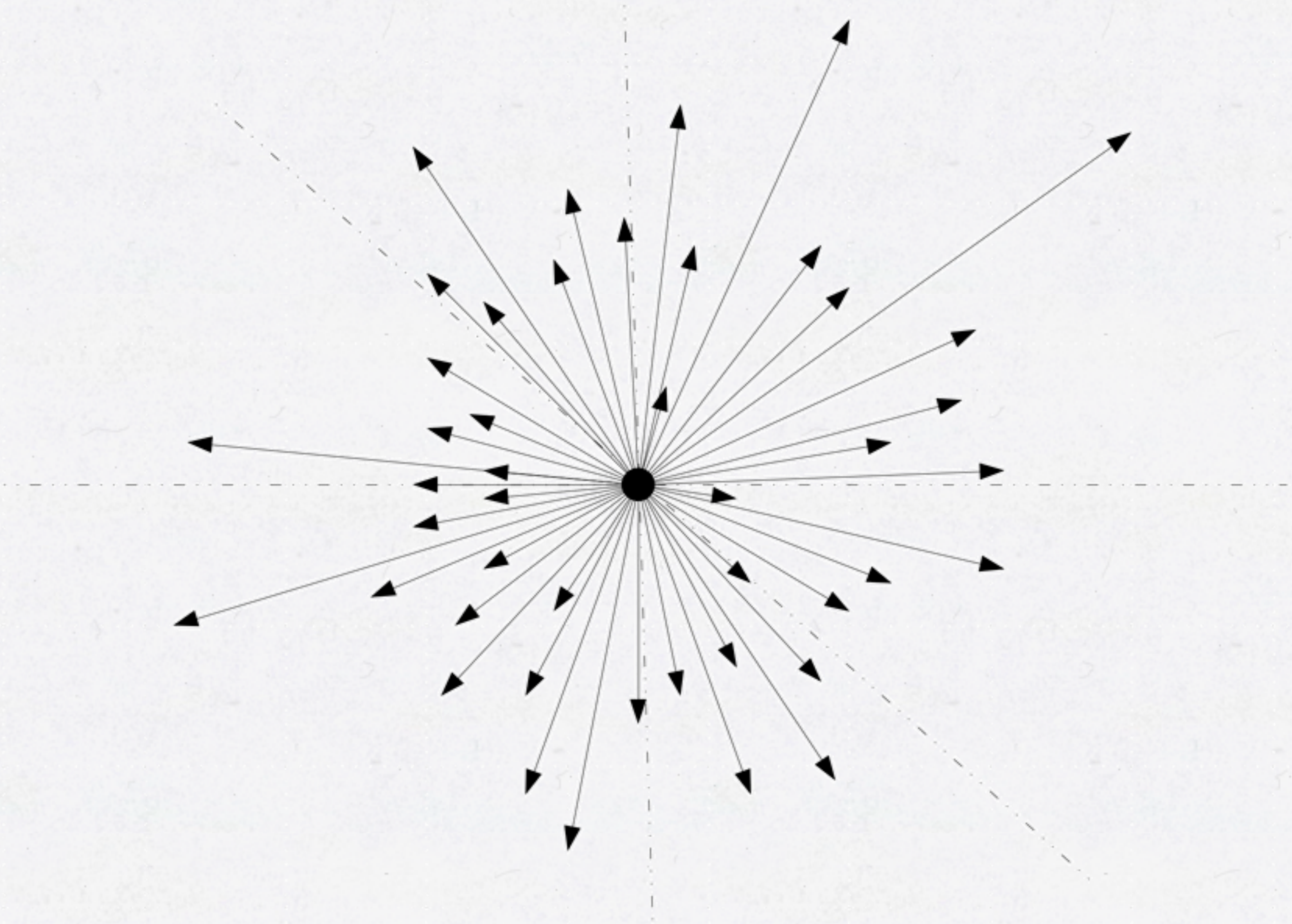
FK_DepartmentID_DepartmentID
FK_DepartmentHeadID_EmployeeID
FK_Code_Code
FK_SalesRepresentative_EmployeeID
FK_FinancialCode_Code
FK_CustomerID_ID
FK_CustomerID_ID
FK_ID_ID
FK_ProductID_ID



CHALLENGE #3: BIAS VS. VARIANCE

BIAS vs. VARIANCE

CHALLENGE #4: VECTORIZATION



HOW DOES JAVA FACE UP TO THOSE CHALLENGES?

FACING CHALLENGE #1: COMPLEXITY

- Ease-of-use:
  - Straight-forward integration with any Java project
  - Same builder pattern used throughout framework
- Tutorials and documentation
- Pre-packaged algorithms (and even ML-models)

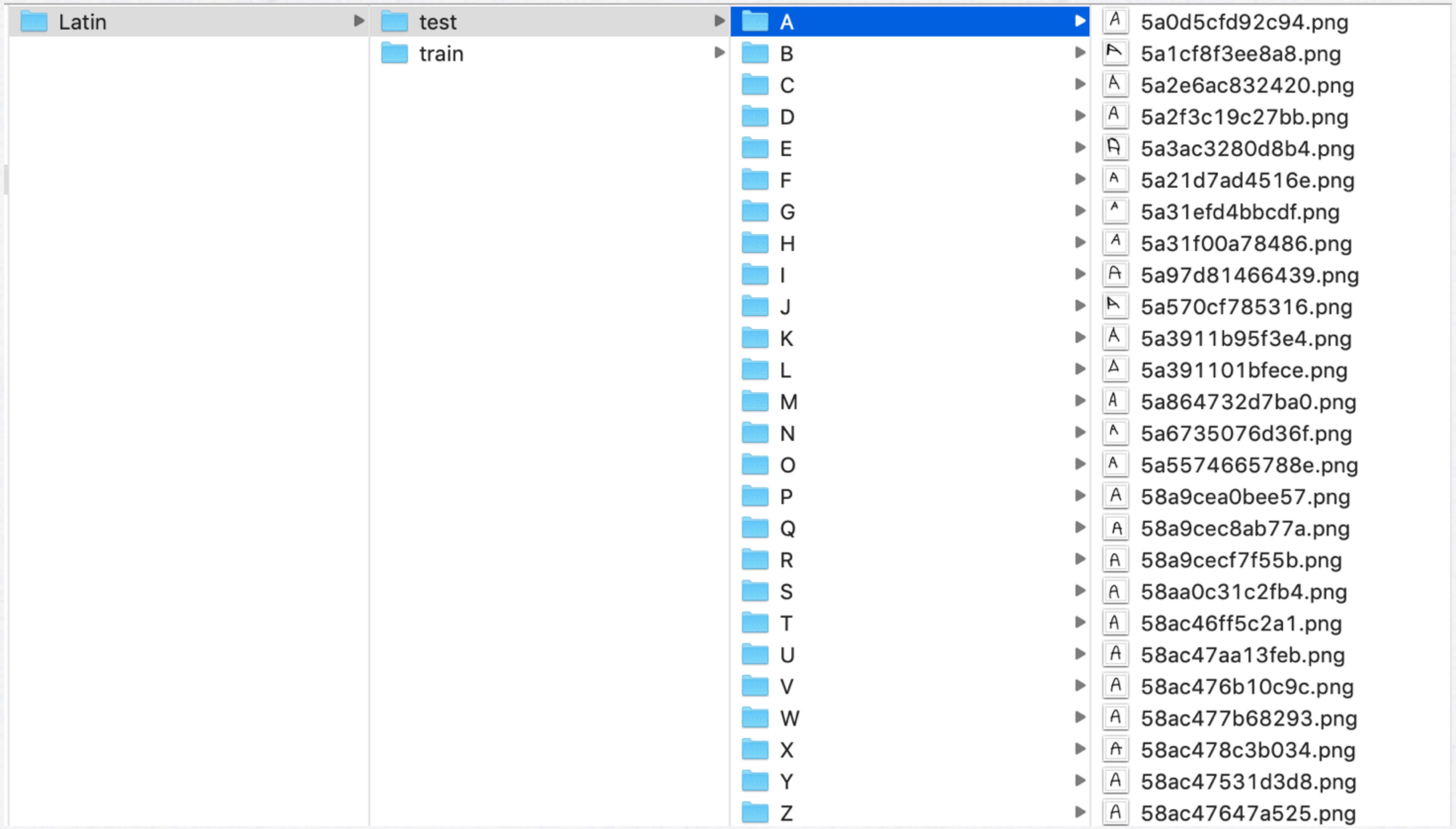
Disclaimer: Also be aware that in order to be successful in implementing machine learning algorithms you probably need more understanding than what a simple tutorial or even a wiki can provide. I strongly suggest spending some time studying the subject before running head-long into implementing intricate machine learning models.

DAVID'S EASY-TO-USE CHEATSHEET

- Model: the algorithm, or program, that makes the prediction/classification/grouping of the input data. I.e. the machine in machine learning.
  - In deep learning the model is called an artificial neural network. There are three basic types:
    - Multilayer perceptron
    - Convolutional neural network
    - Recurrent neural network
- Data can be labelled or unlabelled, and structured or unstructured
  - If data is labelled we can train the model towards a known target, otherwise not
  - If the data is structured it means we know (some of) the features of the data, otherwise not

FACING CHALLENGE #2: DATA

```
FileSplit fileSplit = new FileSplit(directory, {".png"});
ParentPathLabelGenerator labelMaker = new ParentPathLabelGenerator();
ImageRecordReader recordReader = new ImageRecordReader(28,28,1, labelMaker);
recordReader.initialize(fileSplit);
```



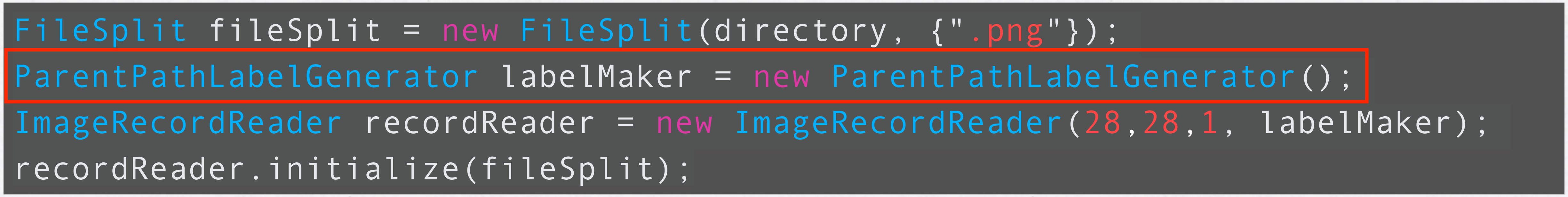
Latin

test
train

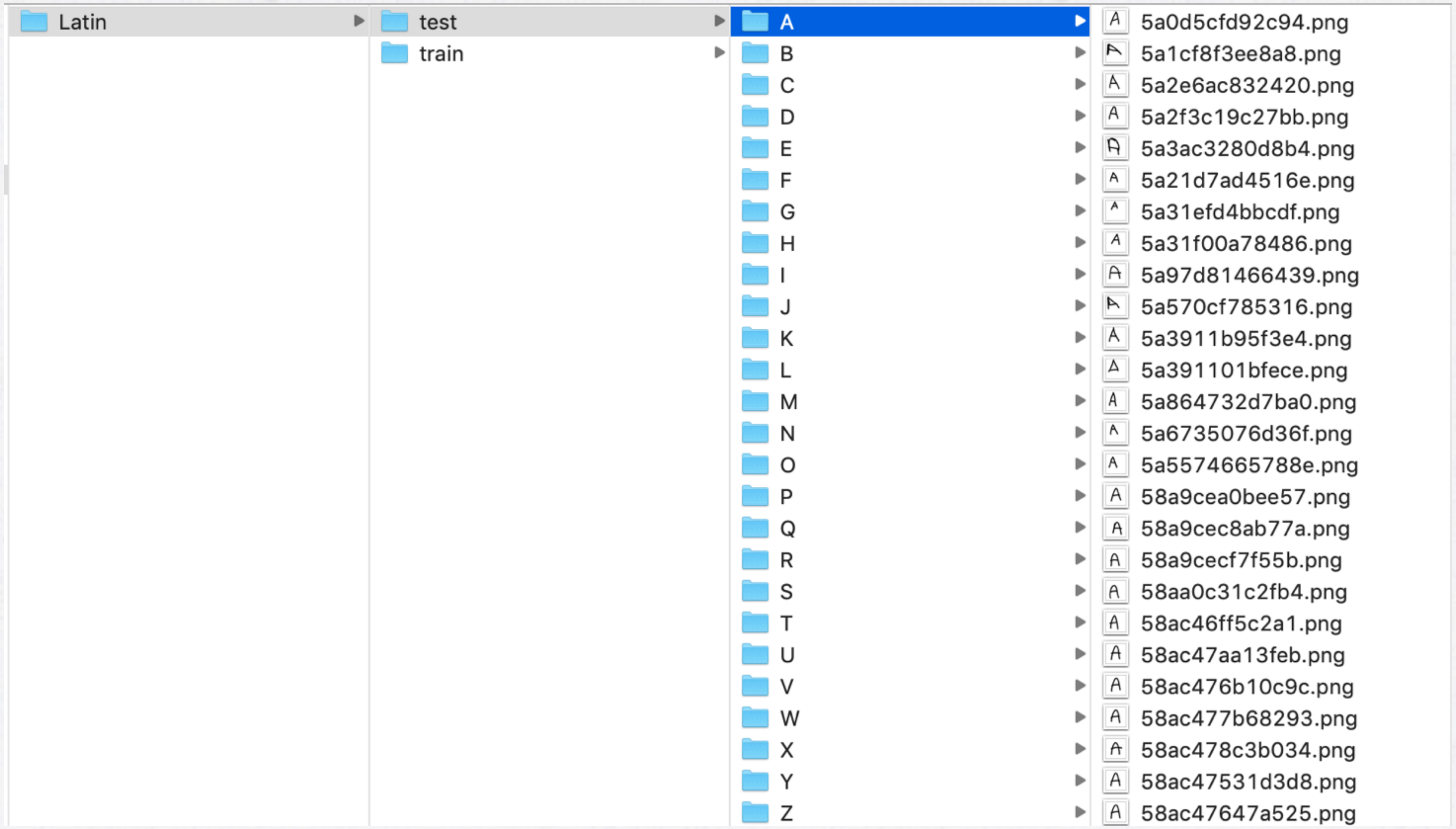
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

5a0d5cfd92c94.png
5a1cf8f3ee8a8.png
5a2e6ac832420.png
5a2f3c19c27bb.png
5a3ac3280d8b4.png
5a21d7ad4516e.png
5a31efd4bbcdf.png
5a31f00a78486.png
5a97d81466439.png
5a570cf785316.png
5a3911b95f3e4.png
5a391101bfece.png
5a864732d7ba0.png
5a6735076d36f.png
5a5574665788e.png
58a9cea0bee57.png
58a9cec8ab77a.png
58a9cecf7f55b.png
58aa0c31c2fb4.png
58ac46ff5c2a1.png
58ac47aa13feb.png
58ac476b10c9c.png
58ac477b68293.png
58ac478c3b034.png
58ac47531d3d8.png
58ac47647a525.png

FACING CHALLENGE #2: DATA

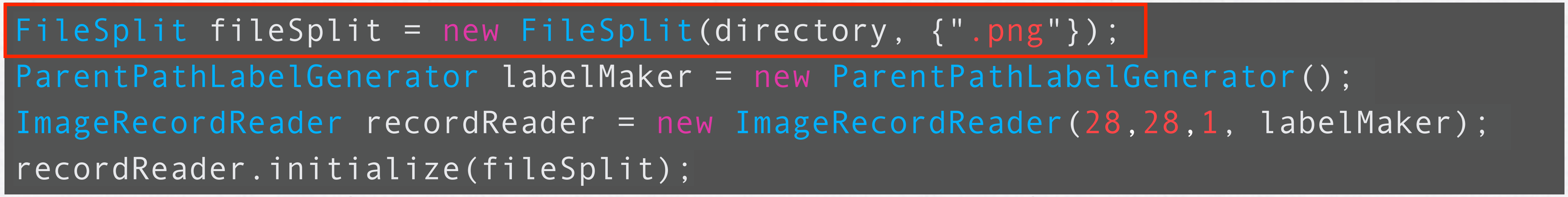


```
FileSplit fileSplit = new FileSplit(directory, {".png"});
ParentPathLabelGenerator labelMaker = new ParentPathLabelGenerator();
ImageRecordReader recordReader = new ImageRecordReader(28,28,1, labelMaker);
recordReader.initialize(fileSplit);
```

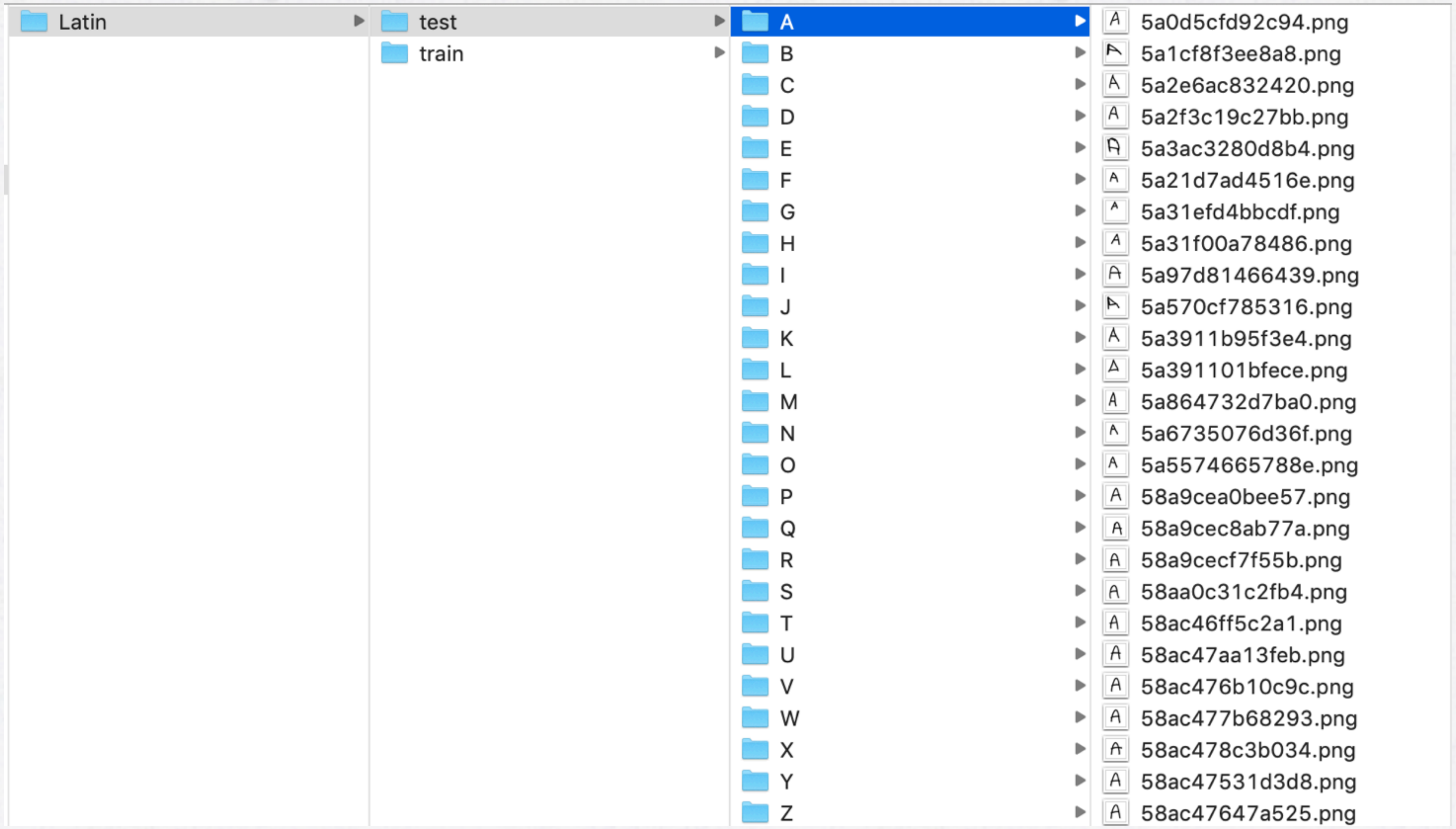


```
Latin ▸   test ▸    A ▸   5a0d5cfd92c94.png
          train ▸   B ▸   5a1cf8f3ee8a8.png
                    C ▸   5a2e6ac832420.png
                    D ▸   5a2f3c19c27bb.png
                    E ▸   5a3ac3280d8b4.png
                    F ▸   5a21d7ad4516e.png
                    G ▸   5a31efd4bbcdf.png
                    H ▸   5a31f00a78486.png
                    I ▸   5a97d81466439.png
                    J ▸   5a570cf785316.png
                    K ▸   5a3911b95f3e4.png
                    L ▸   5a391101bfece.png
                    M ▸   5a864732d7ba0.png
                    N ▸   5a6735076d36f.png
                    O ▸   5a5574665788e.png
                    P ▸   58a9cea0bee57.png
                    Q ▸   58a9cec8ab77a.png
                    R ▸   58a9cecf7f55b.png
                    S ▸   58aa0c31c2fb4.png
                    T ▸   58ac46ff5c2a1.png
                    U ▸   58ac47aa13feb.png
                    V ▸   58ac476b10c9c.png
                    W ▸   58ac477b68293.png
                    X ▸   58ac478c3b034.png
                    Y ▸   58ac47531d3d8.png
                    Z ▸   58ac47647a525.png
```

FACING CHALLENGE #2: DATA



```
FileSplit fileSplit = new FileSplit(directory, {".png"});
ParentPathLabelGenerator labelMaker = new ParentPathLabelGenerator();
ImageRecordReader recordReader = new ImageRecordReader(28,28,1, labelMaker);
recordReader.initialize(fileSplit);
```



Latin

test
train

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

5a0d5cfd92c94.png
5a1cf8f3ee8a8.png
5a2e6ac832420.png
5a2f3c19c27bb.png
5a3ac3280d8b4.png
5a21d7ad4516e.png
5a31efd4bbcdf.png
5a31f00a78486.png
5a97d81466439.png
5a570cf785316.png
5a3911b95f3e4.png
5a391101bfece.png
5a864732d7ba0.png
5a6735076d36f.png
5a5574665788e.png
58a9cea0bee57.png
58a9cec8ab77a.png
58a9cecf7f55b.png
58aa0c31c2fb4.png
58ac46ff5c2a1.png
58ac47aa13feb.png
58ac476b10c9c.png
58ac477b68293.png
58ac478c3b034.png
58ac47531d3d8.png
58ac47647a525.png

FACING CHALLENGE #3: BIAS VS. VARIANCE

- Arbiter is a tool in the DL4J framework to automatically fine-tune a model's parameter space (learning rate, network size, regularization, etc.)
- Virtually anything in the model configuration can be tuned in
- Drawback: a lot more processing time to fine-tune parameters

FACING CHALLENGE #4: VECTORIZATION

- Vectorization in DL4J is handled by ND4J (n-dimensional array)
- JNI bridge to same C++ libraries used by NumPy
- Allocates continuous blocks of of-heap memory (performance)
- GPU (CUDA)

HOW TO IMPLEMENT ML SOLUTION IN JAVA?

SETTING UP DEEPLEARNING4J

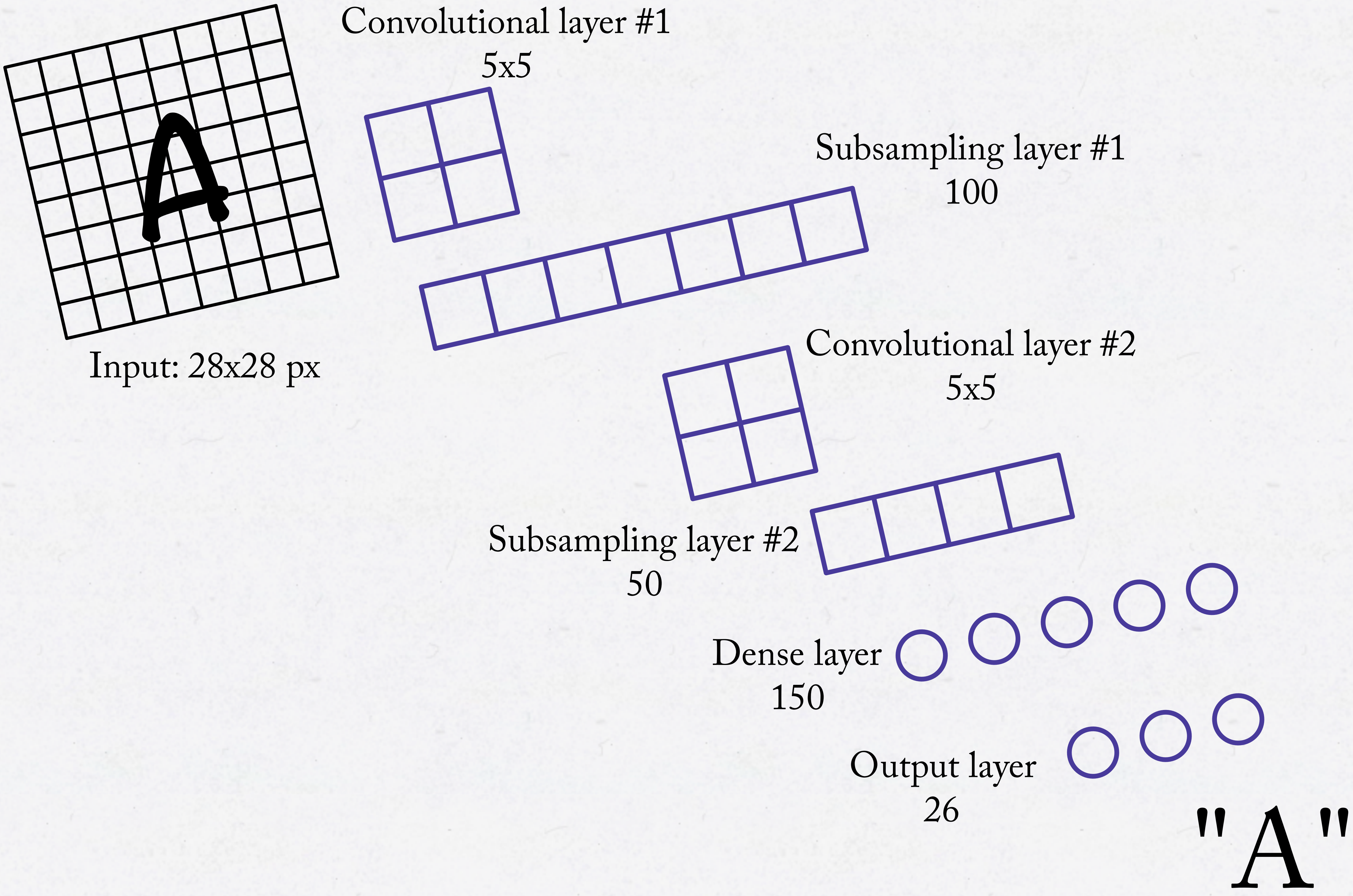
```
dependencies {
  compile("org.deeplearning4j:deeplearning4j-core:1.0.0-beta3")
  compile("org.nd4j:nd4j-native-platform:1.0.0-beta3")
}
```

Alternatively:

```
dependencies {
  compile("org.deeplearning4j:deeplearning4j-core:1.0.0-beta3")
  compile("org.nd4j:nd4j-cuda-9.2:1.0.0-beta3")
}
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK

- We will build a convolutional neural network using Deeplearning4J



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

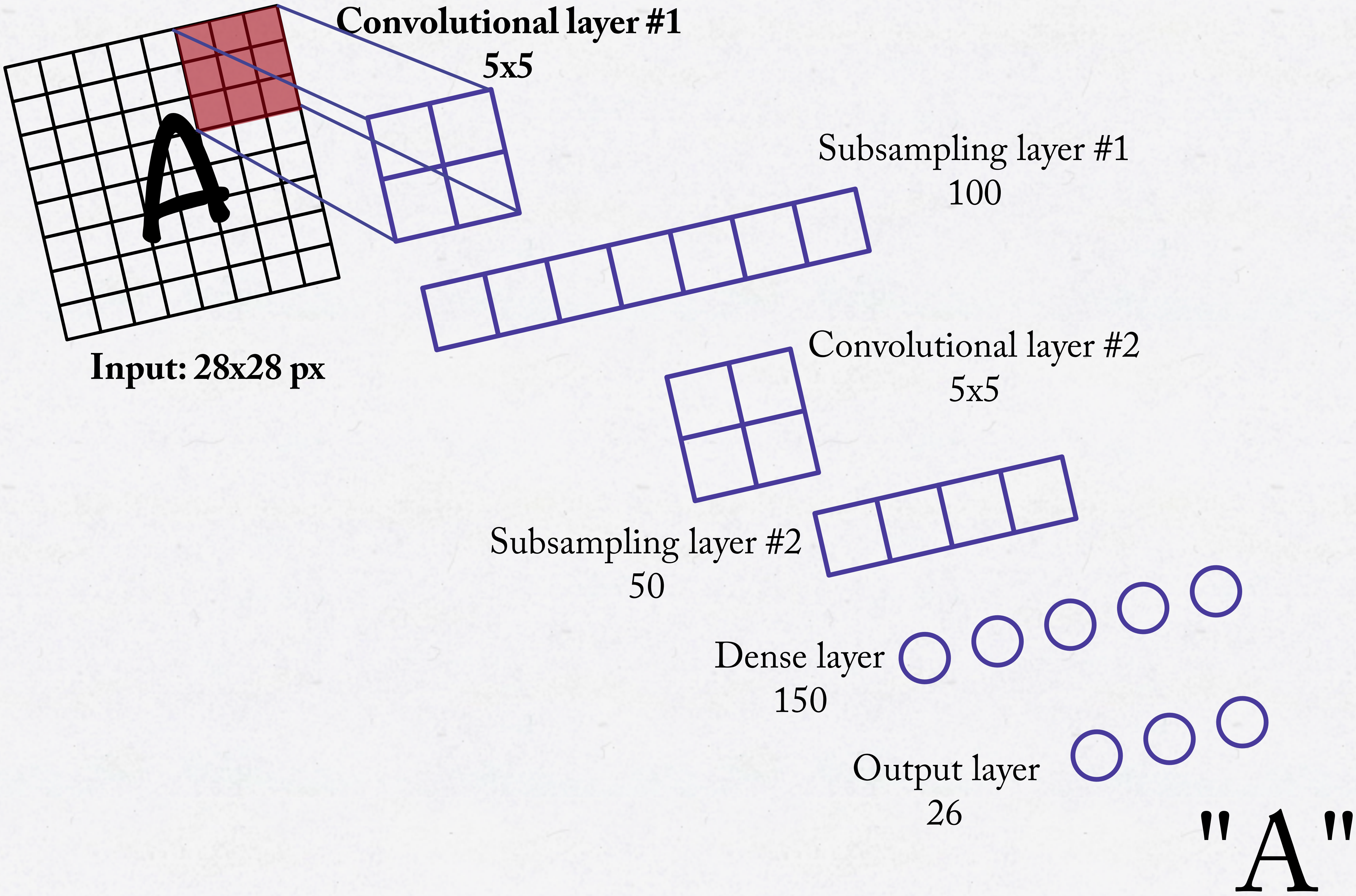
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Input: 28x28 px

Subsampling layer #1
100

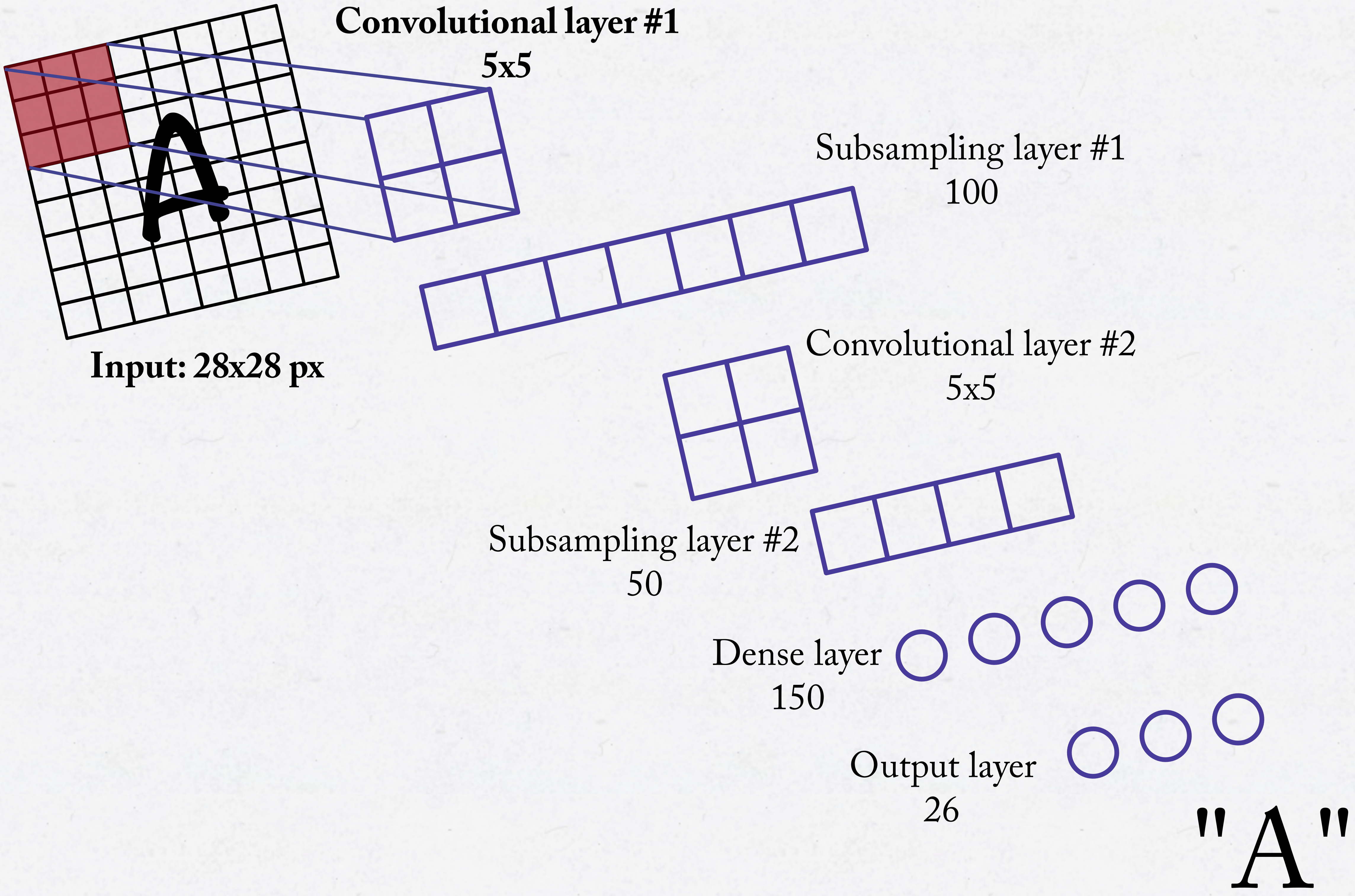
Convolutional layer #2
5x5

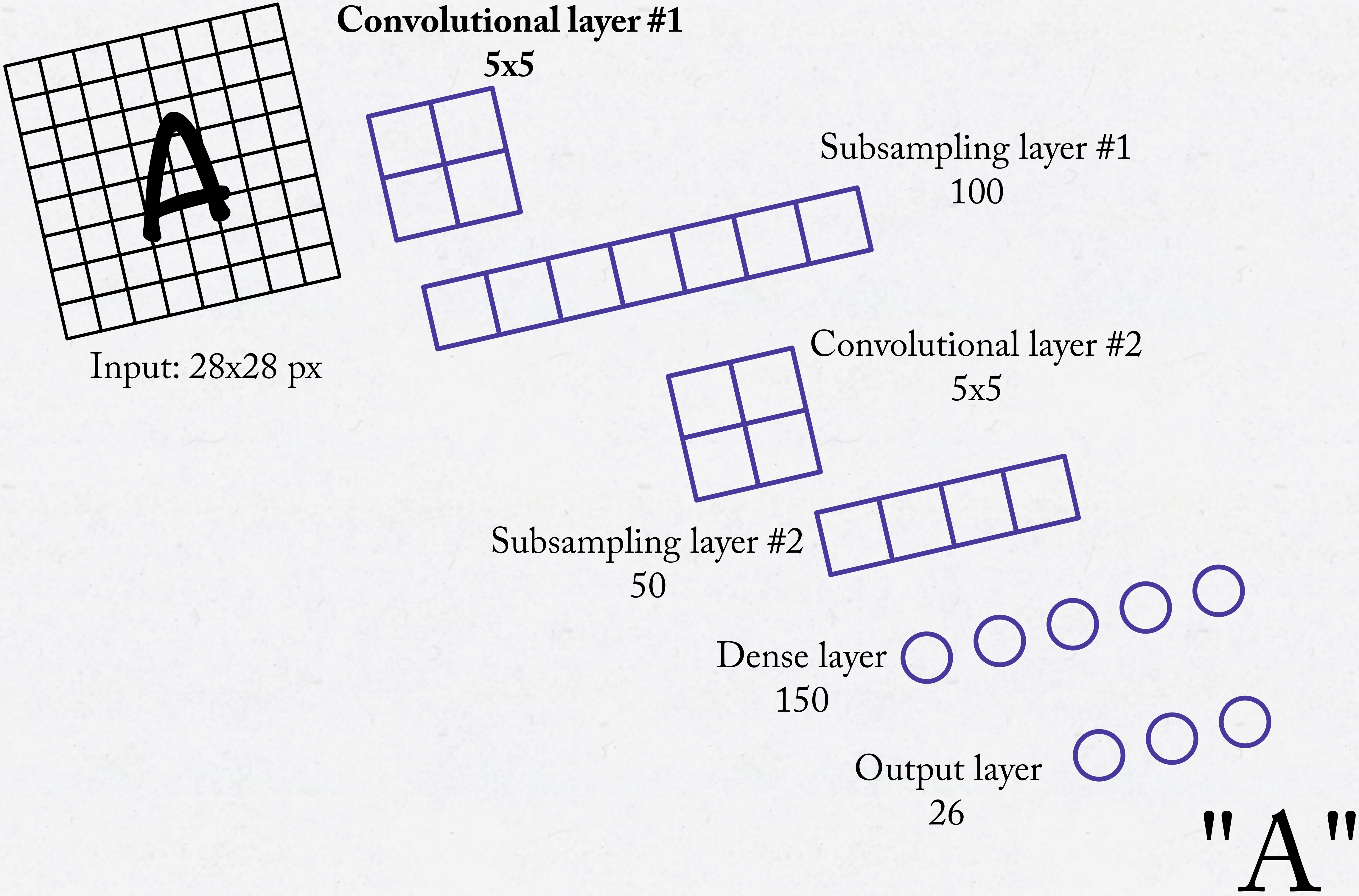
Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"


Convolutional layer #1
5x5
Subsampling layer #1
100
Input: 28x28 px
Convolutional layer #2
5x5
Subsampling layer #2
50
Dense layer
150
Output layer
26
"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

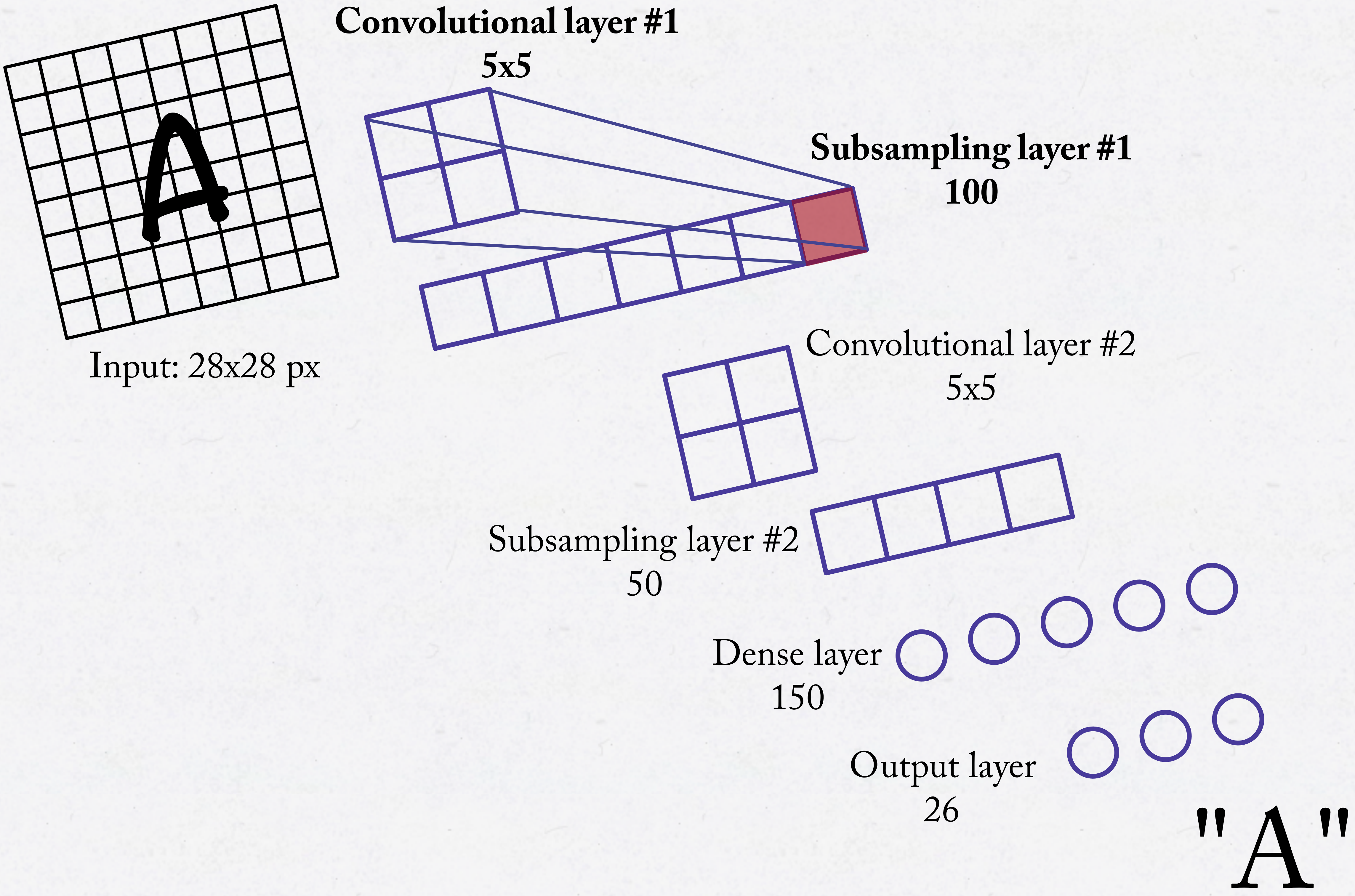
Convolutional layer #2
5x5

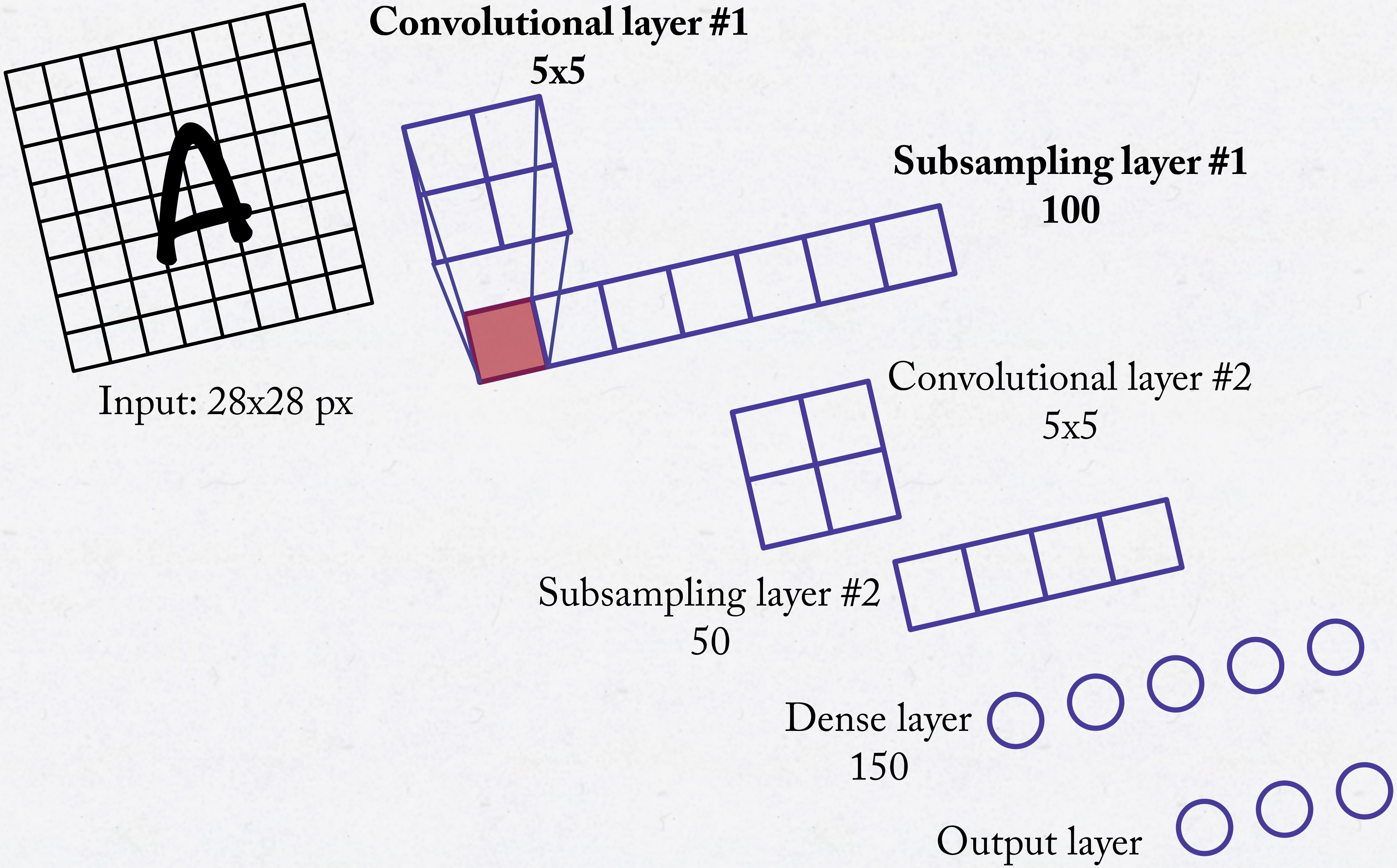
Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"


Convolutional layer #1
5x5
Subsampling layer #1
100
Input: 28x28 px
Convolutional layer #2
5x5
Subsampling layer #2
50
Dense layer
150
Output layer
26
"A"



Input: 28x28 px

Convolutional layer #1
5x5

Subsampling layer #1
100

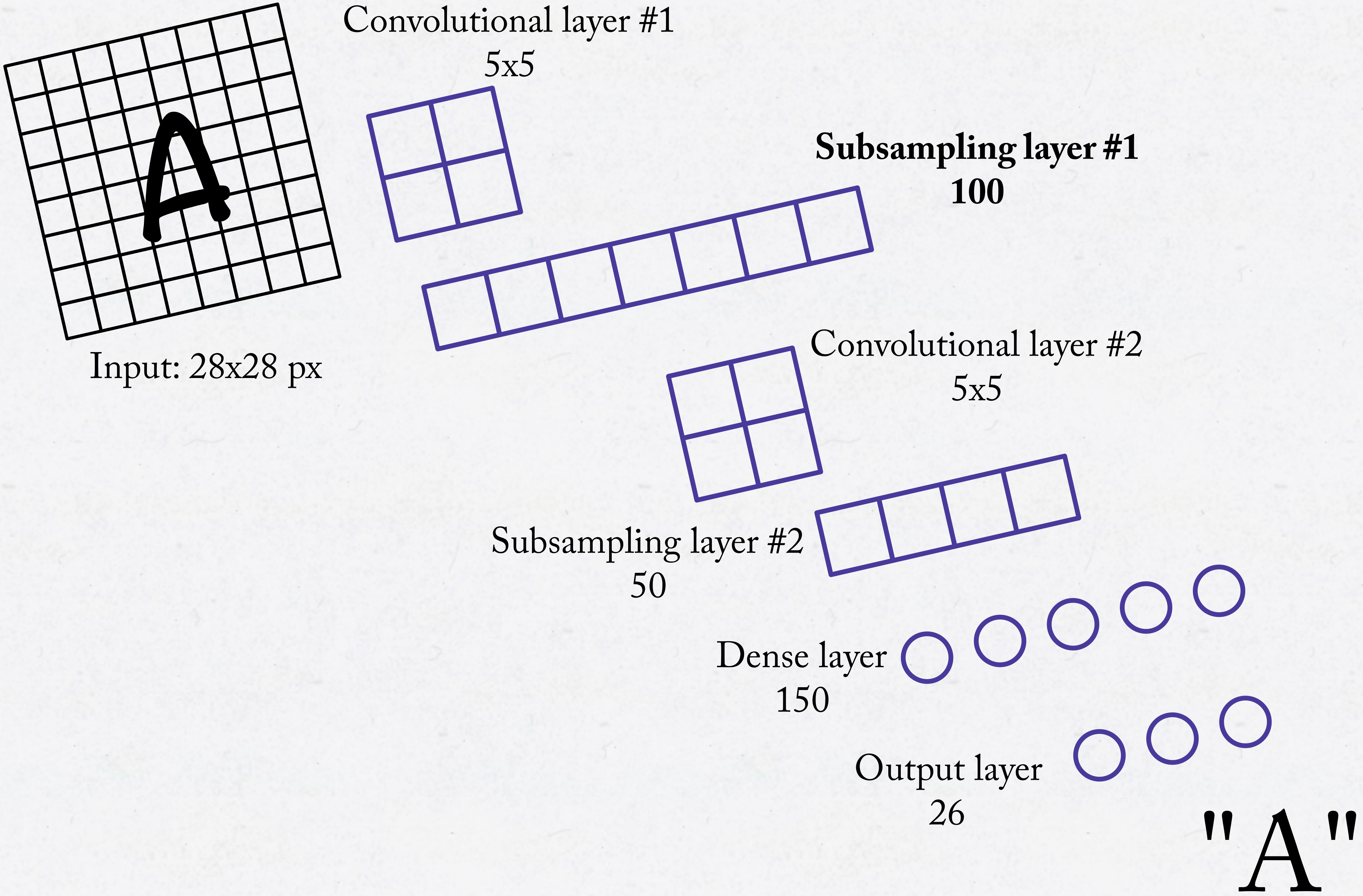
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

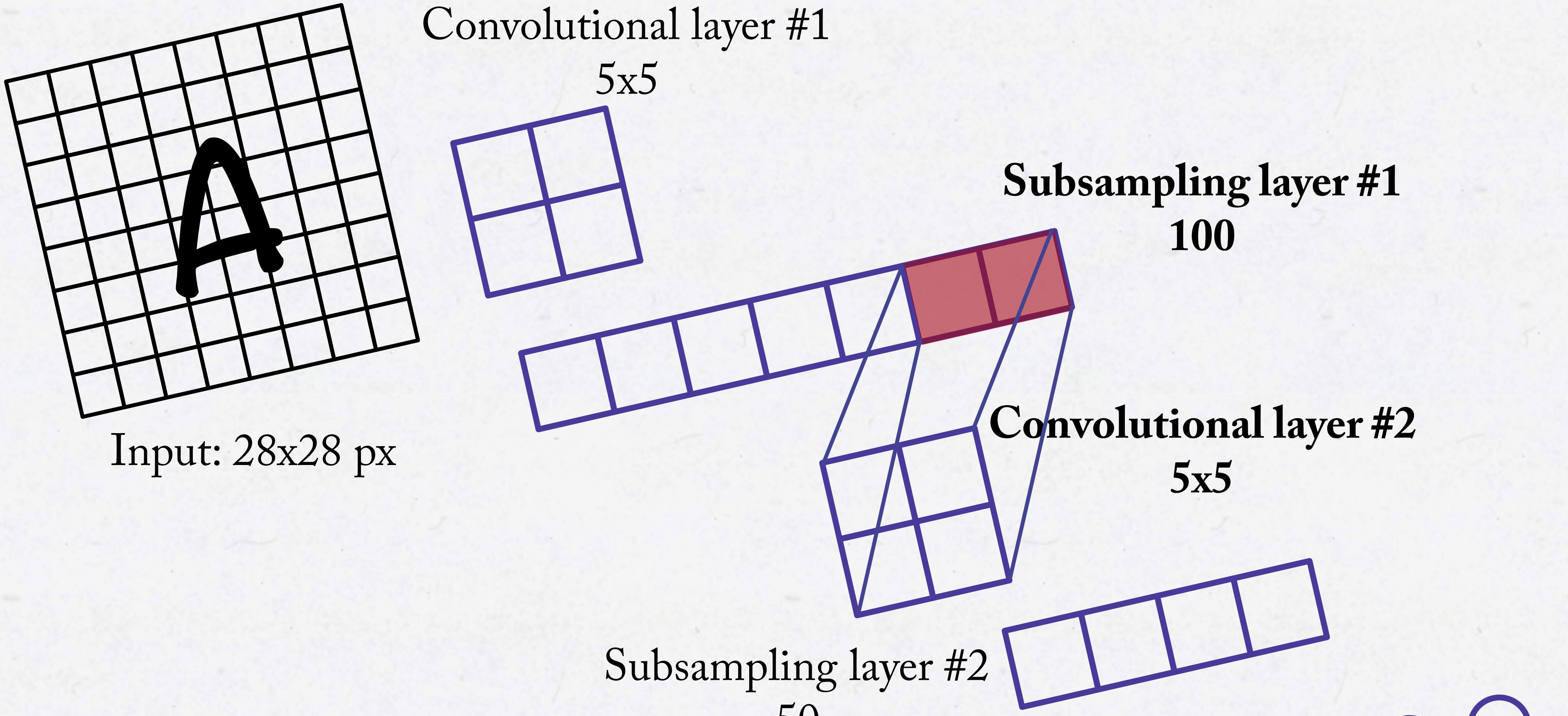
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

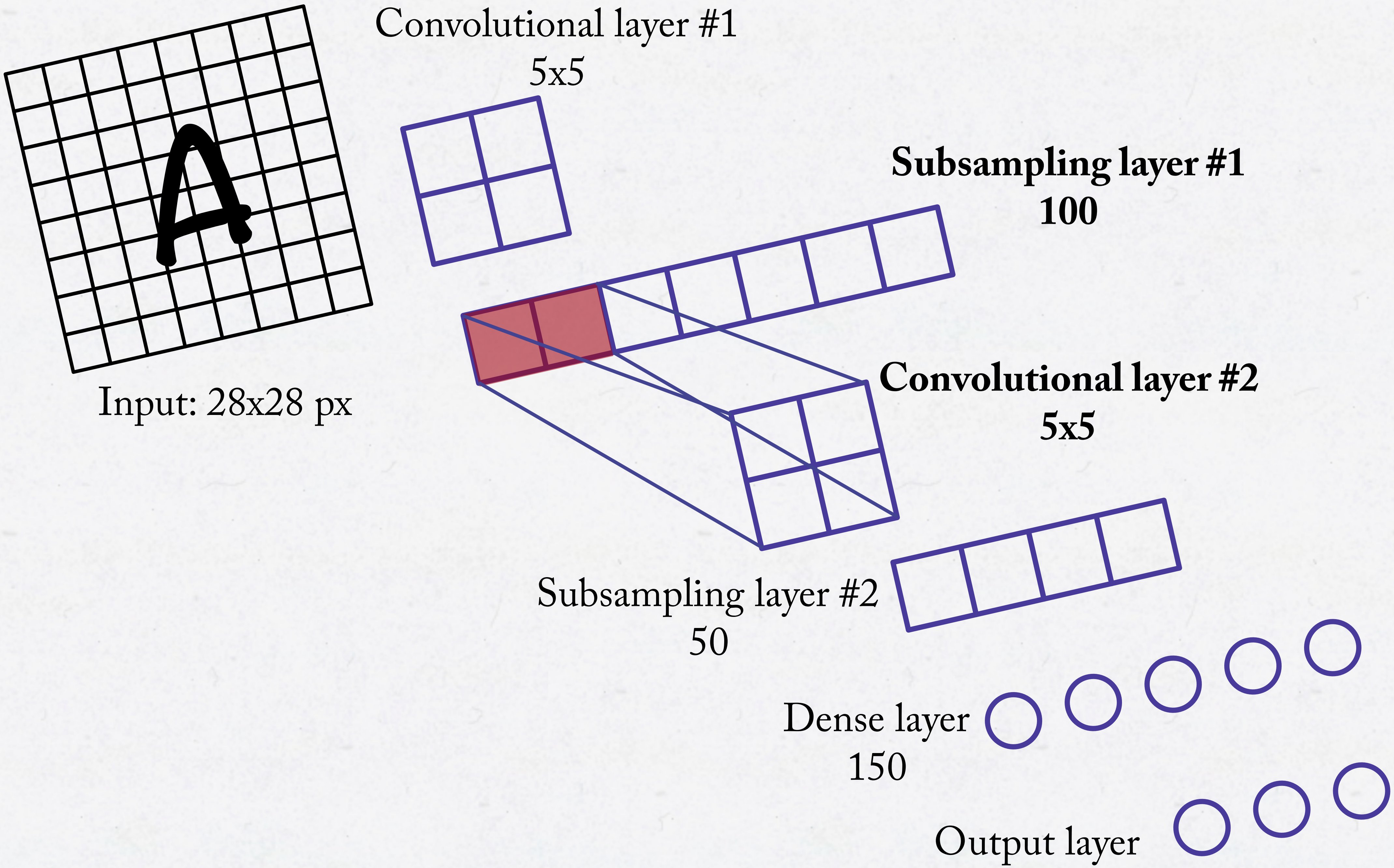
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

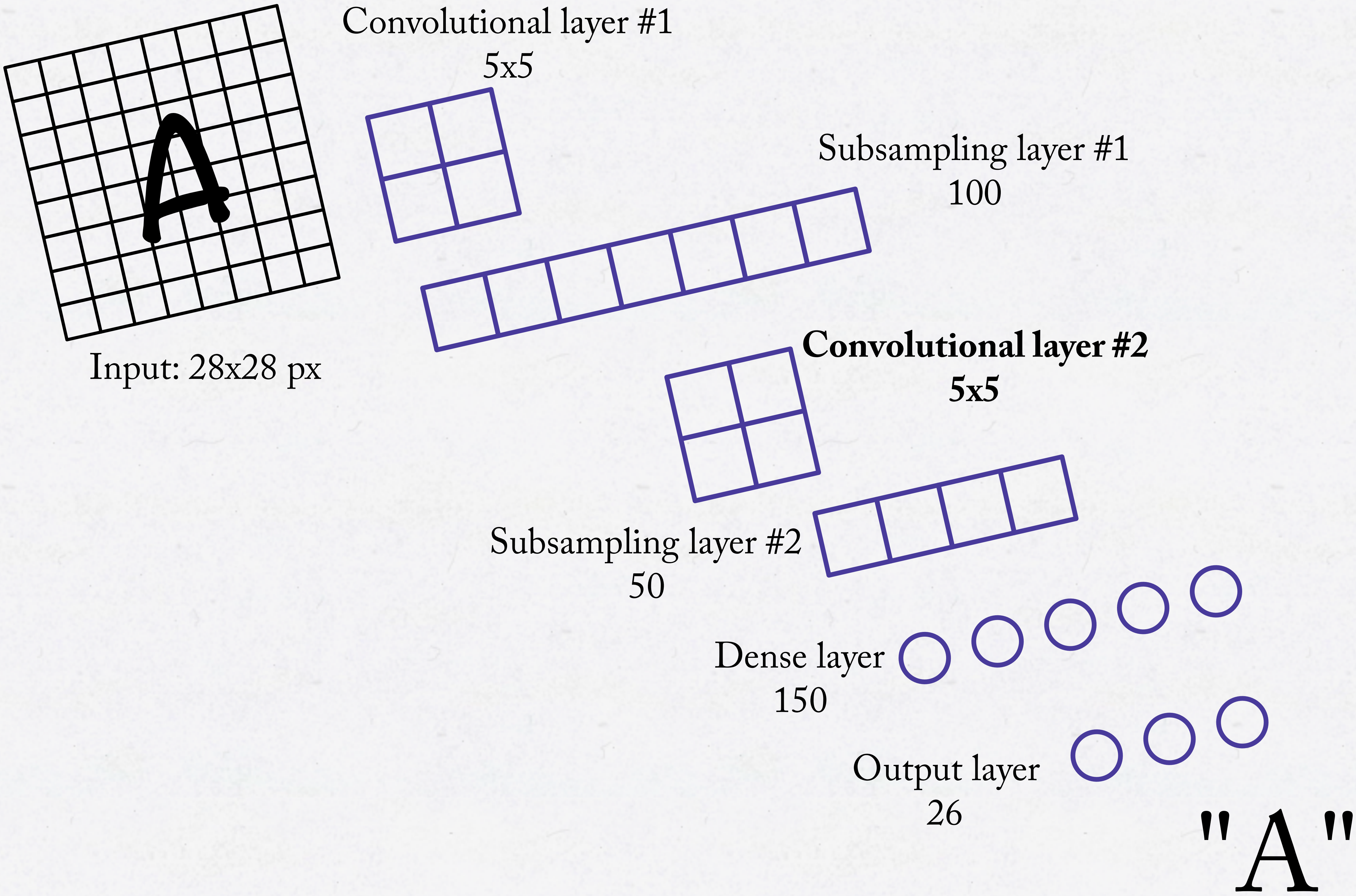
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Input: 28x28 px

Convolutional layer #1
5x5

Subsampling layer #1
100

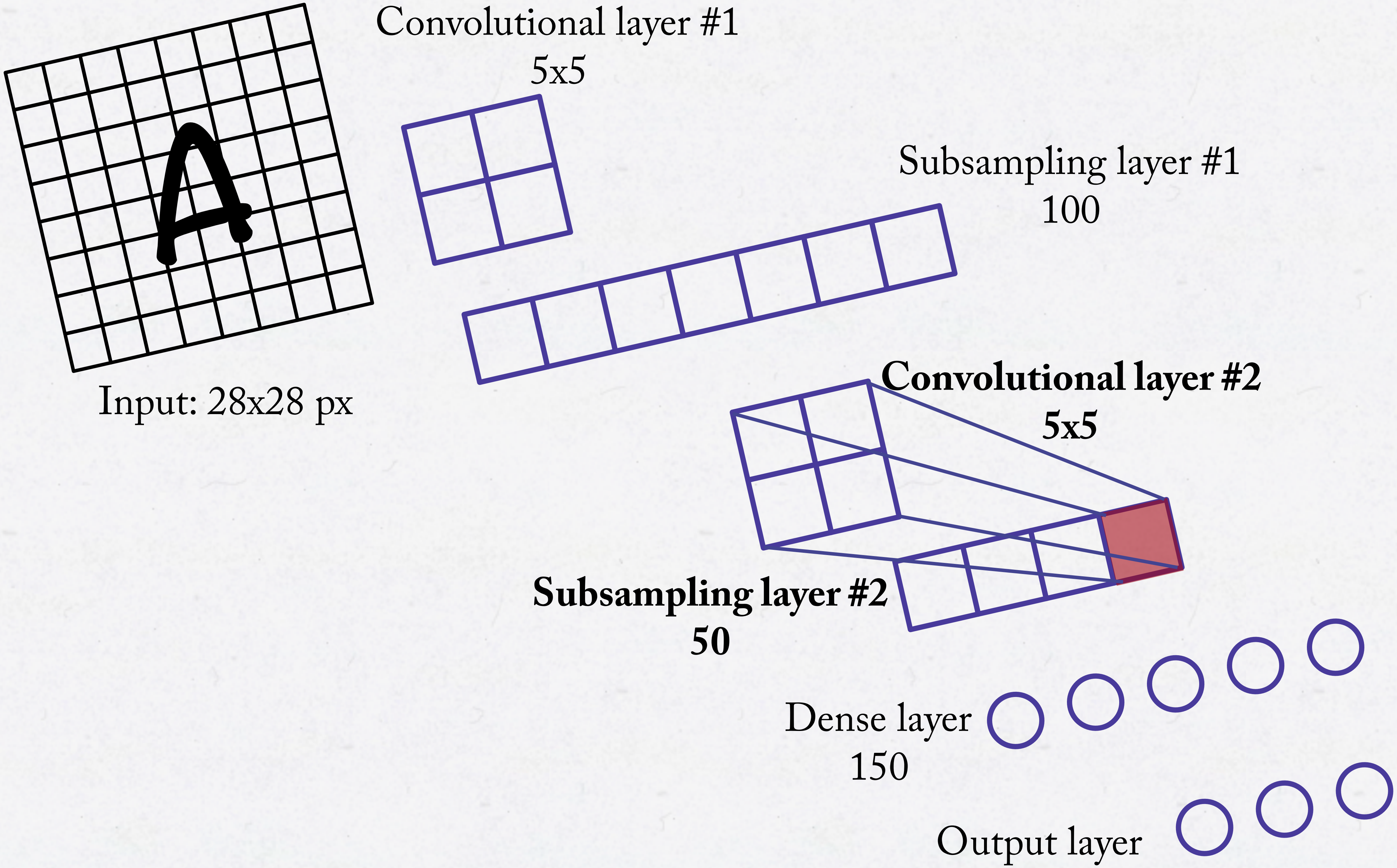
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Input: 28x28 px

Convolutional layer #1
5x5

Subsampling layer #1
100

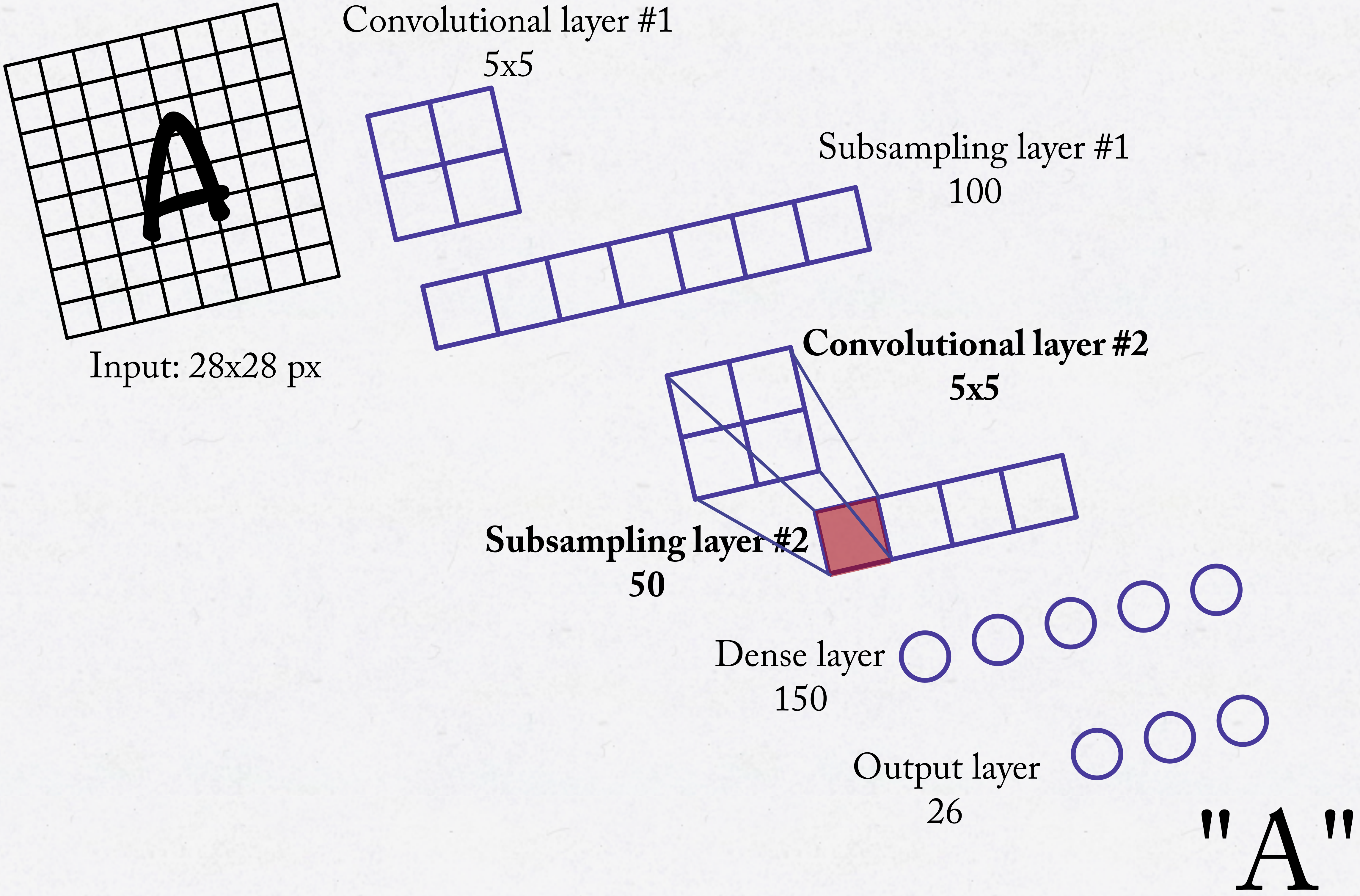
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Input: 28x28 px

Convolutional layer #1
5x5

Subsampling layer #1
100

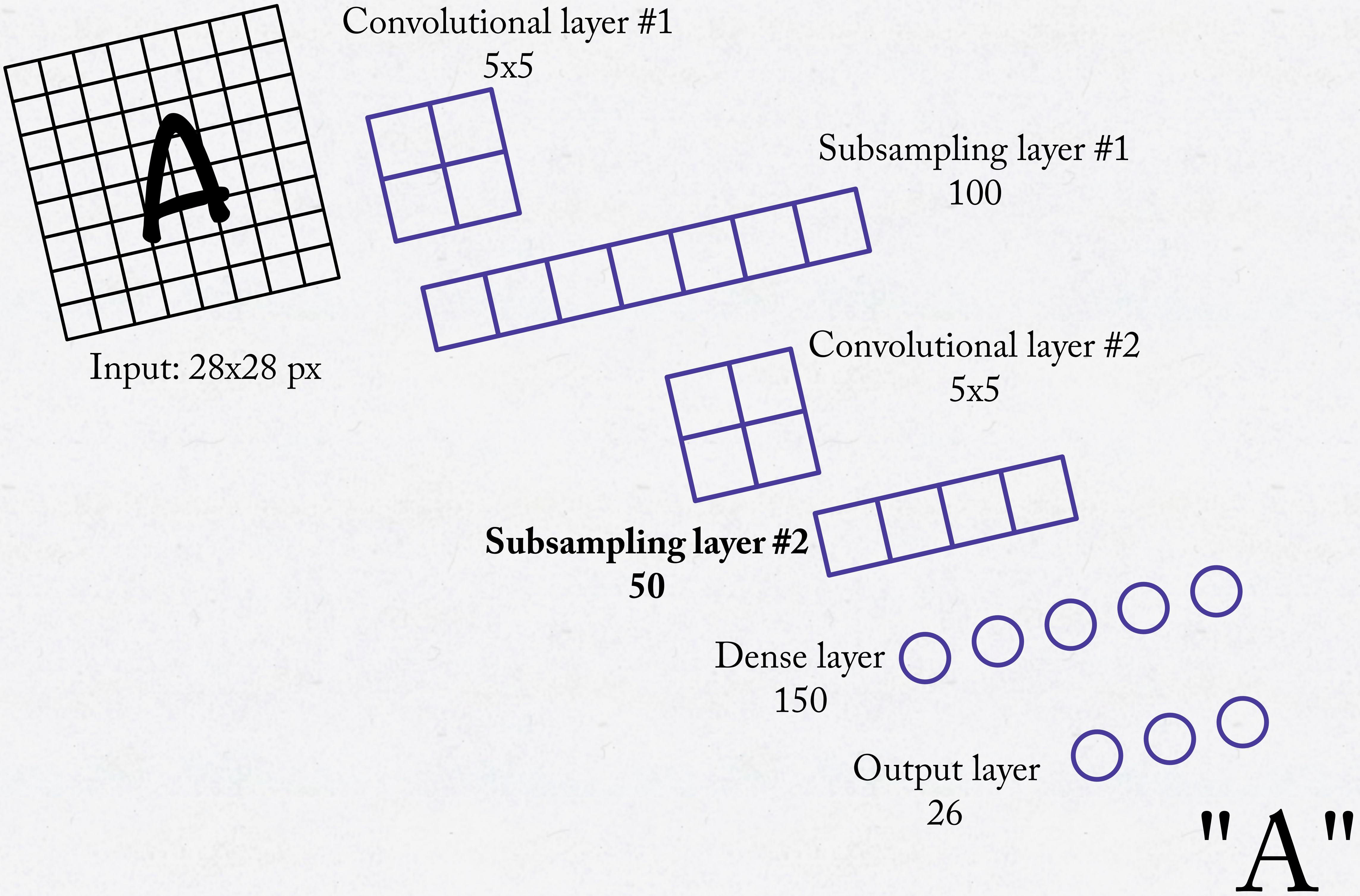
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Input: 28x28 px

Convolutional layer #1
5x5

Subsampling layer #1
100

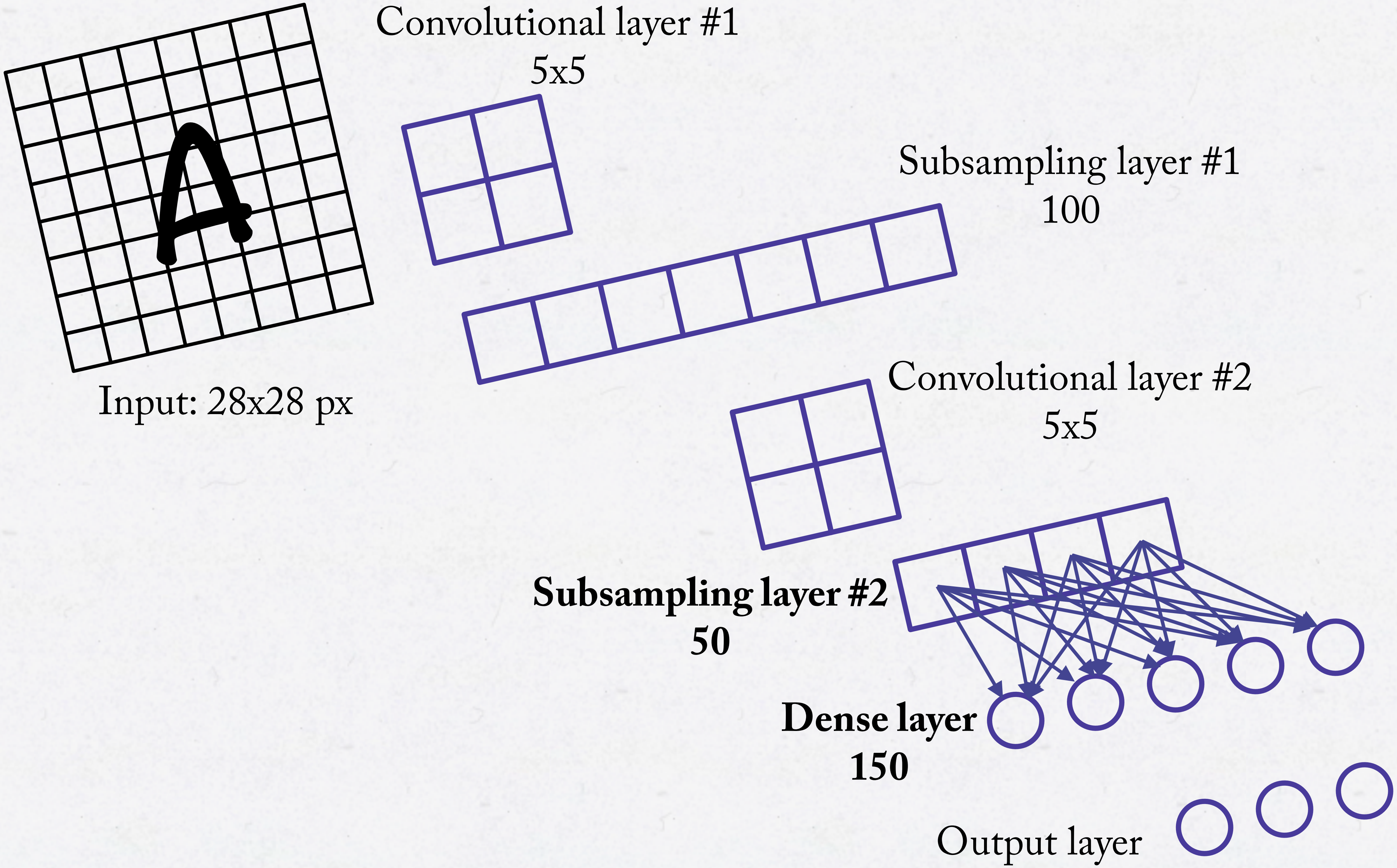
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

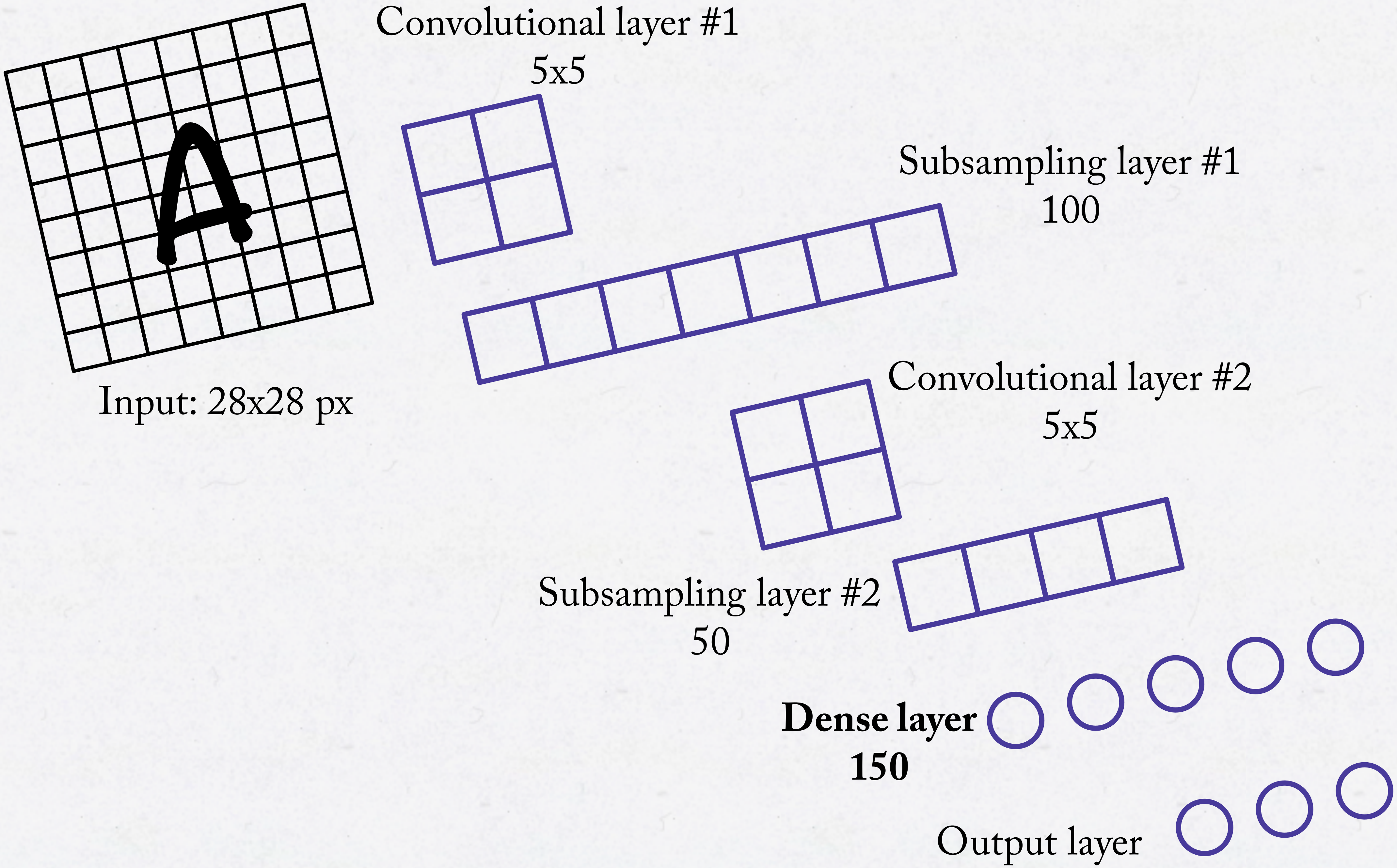
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

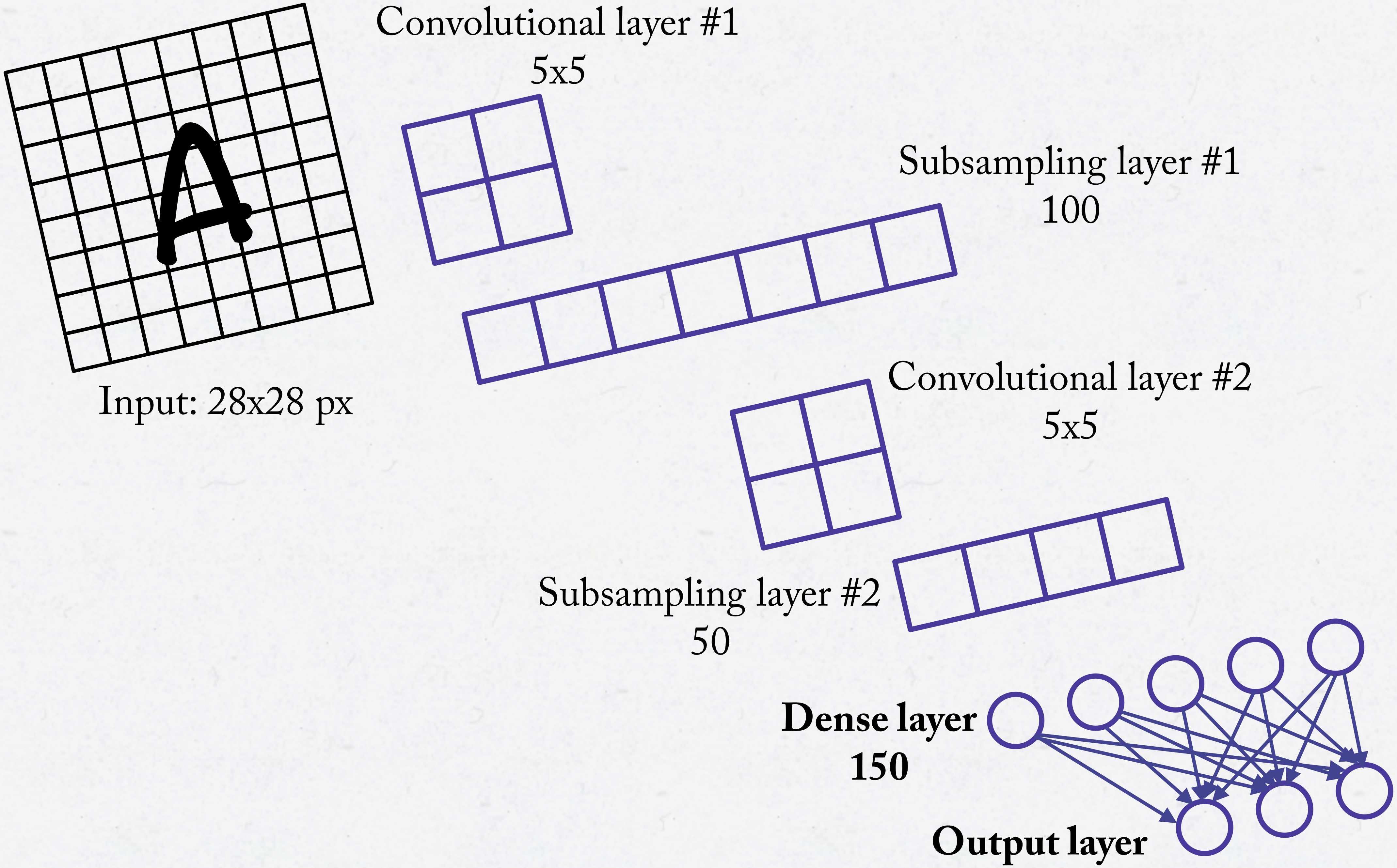
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

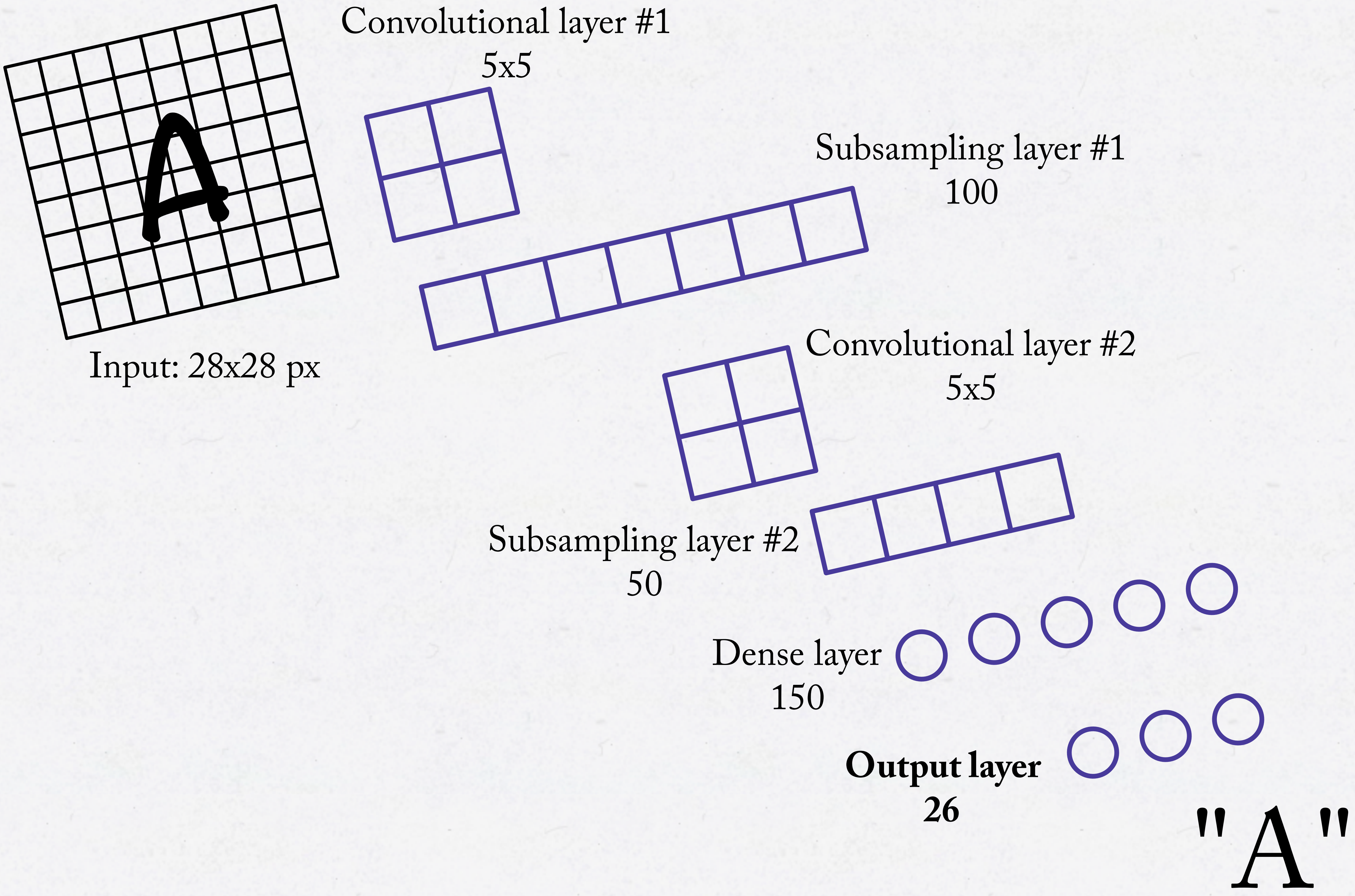
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

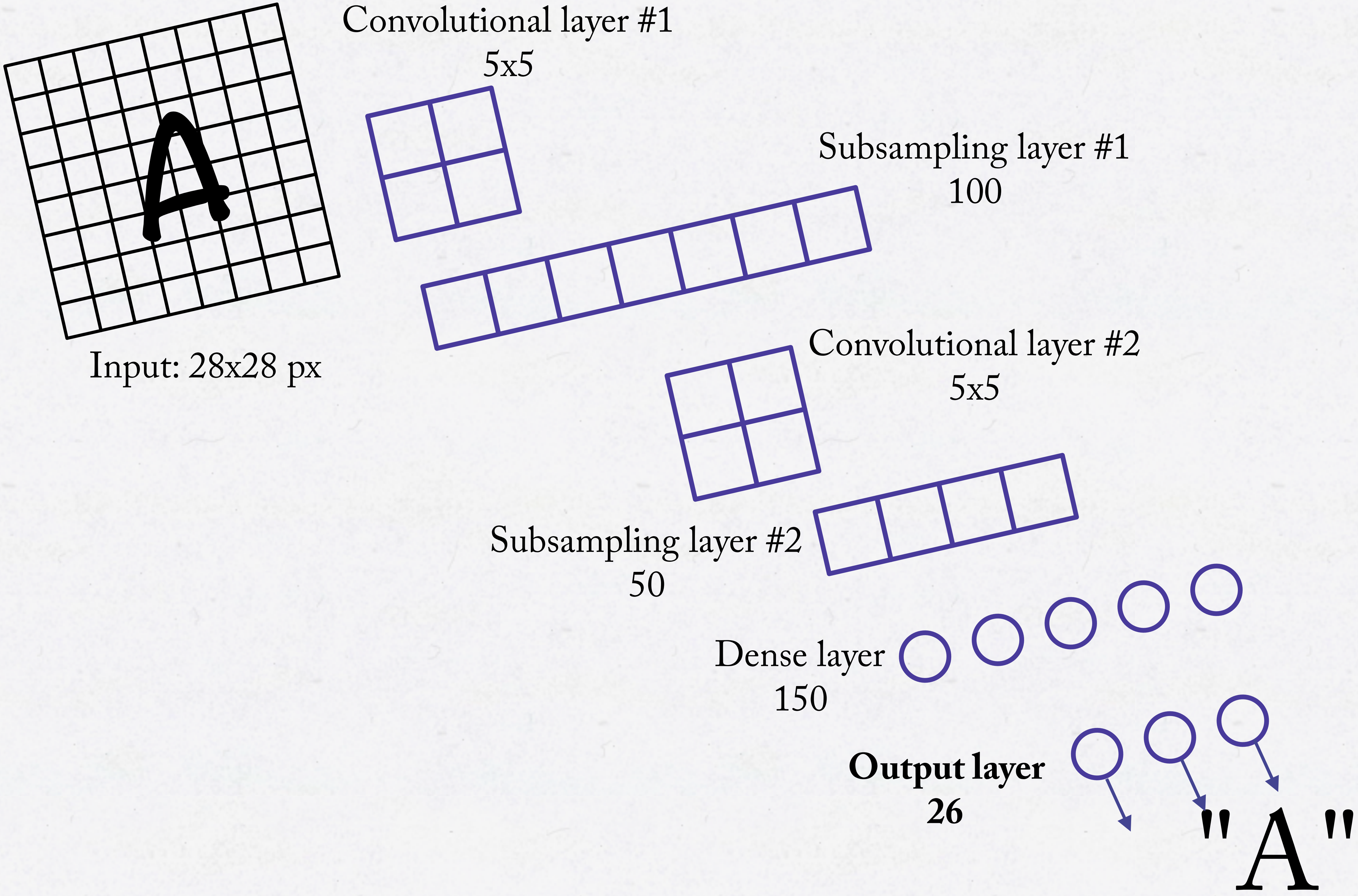
Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

Output layer
26

"A"



Convolutional layer #1
5x5

Subsampling layer #1
100

Input: 28x28 px

Convolutional layer #2
5x5

Subsampling layer #2
50

Dense layer
150

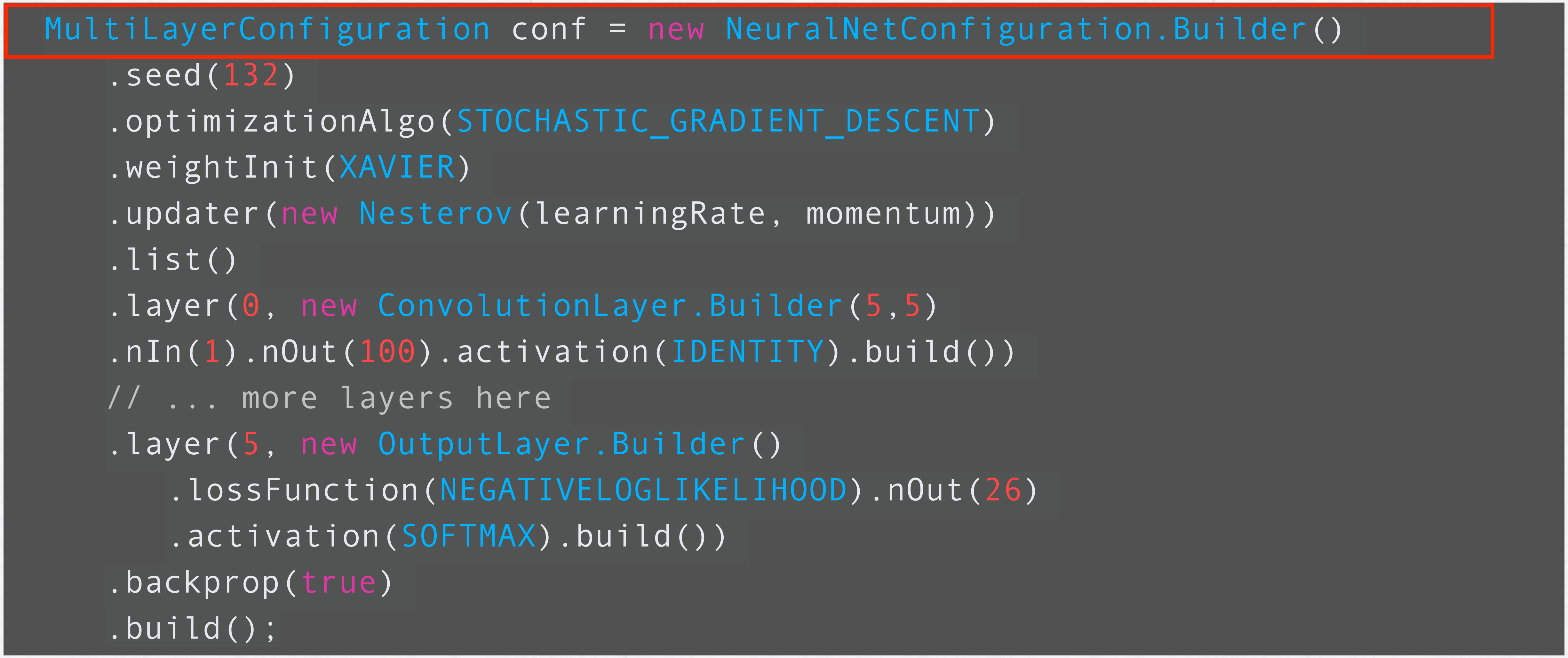
Output layer
26

"A"

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK

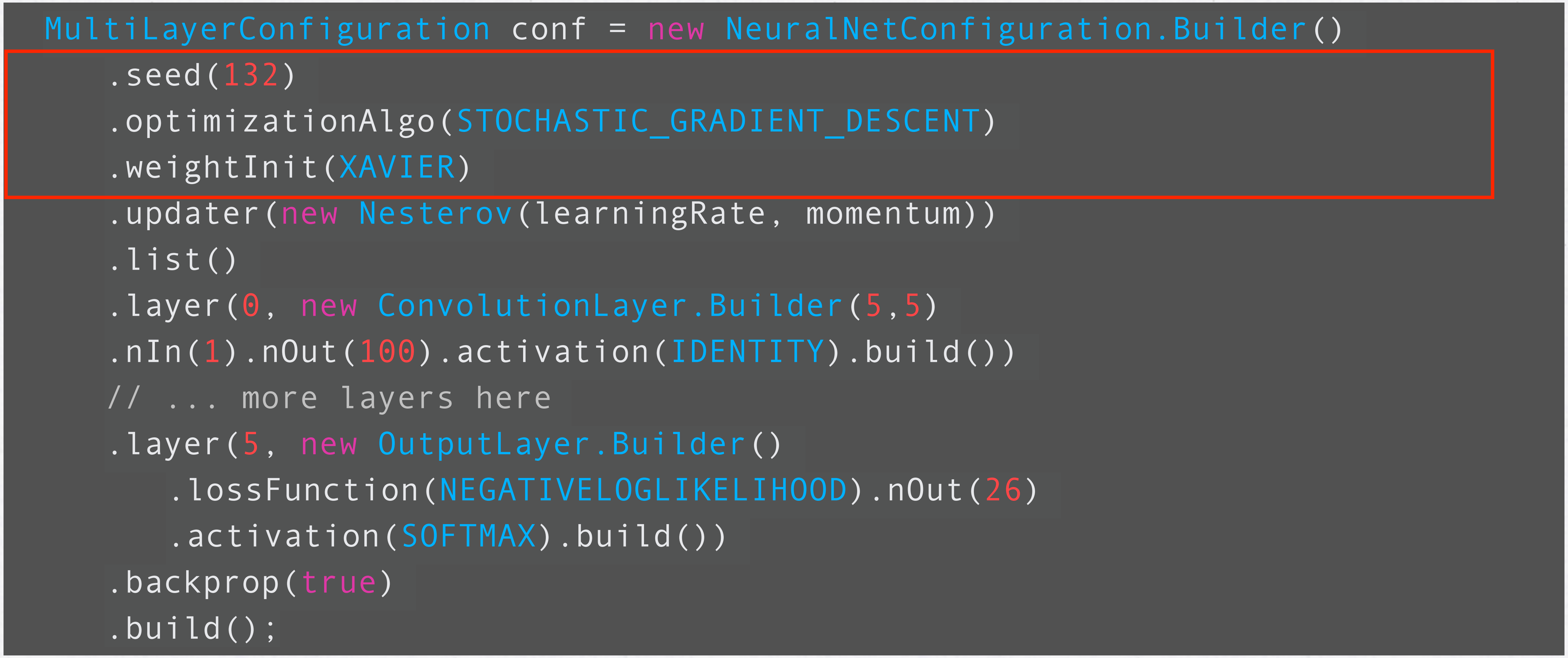
- We will build a convolutional neural network using Deeplearning4J
- Use Deeplearning4J MultilayerConfiguration to build a MultiLayerNetwork

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK



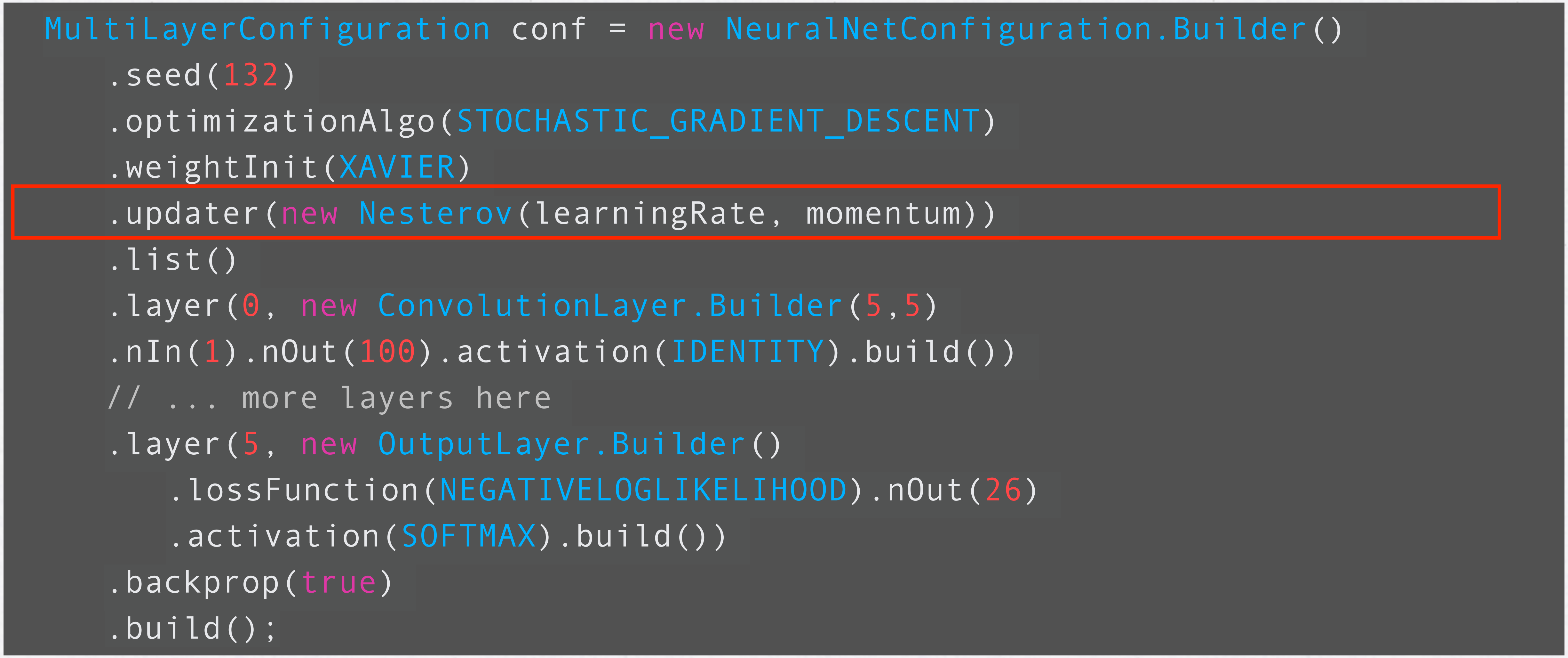
```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(132)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .weightInit(XAVIER)
    .updater(new Nesterov(learningRate, momentum))
    .list()
    .layer(0, new ConvolutionLayer.Builder(5,5)
    .nIn(1).nOut(100).activation(IDENTITY).build())
    // ... more layers here
    .layer(5, new OutputLayer.Builder()
        .lossFunction(NEGATIVELOGLIKELIHOOD).nOut(26)
        .activation(SOFTMAX).build())
    .backprop(true)
    .build();
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK



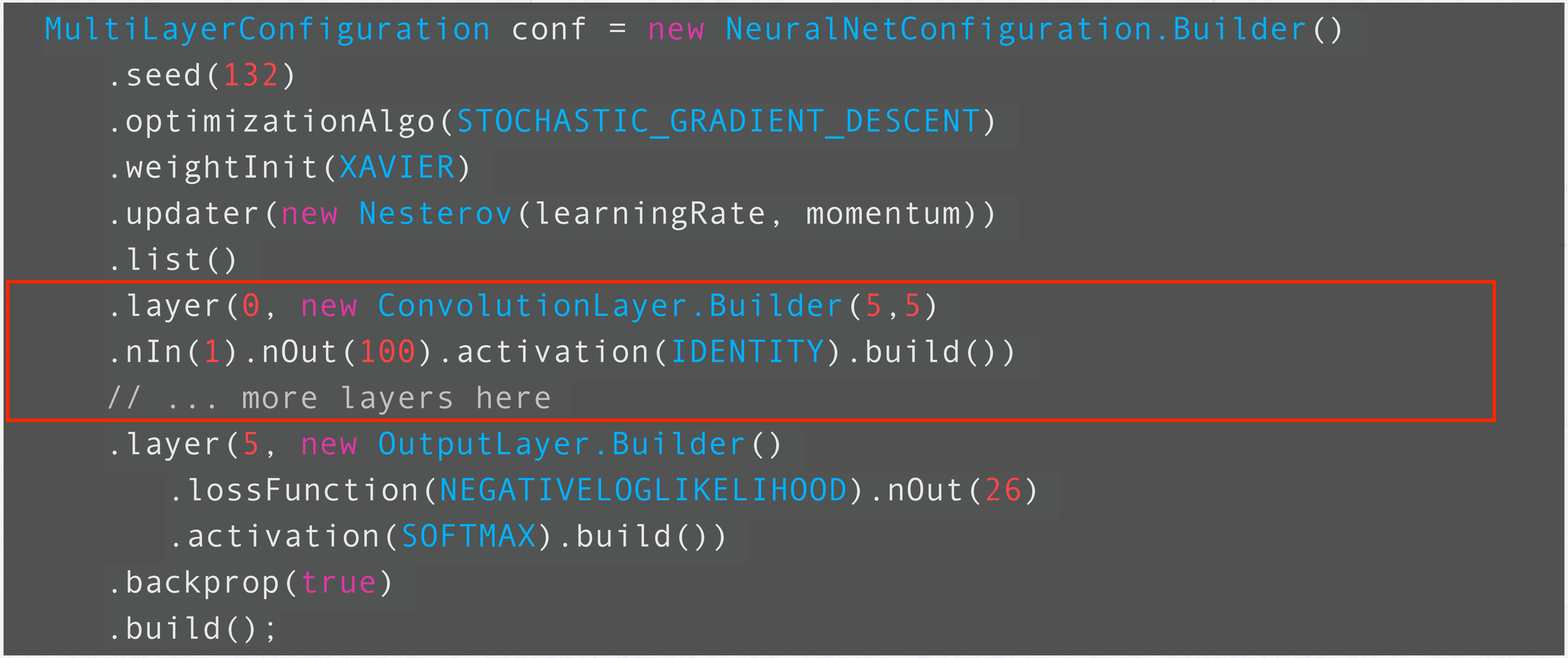
```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(132)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .weightInit(XAVIER)
    .updater(new Nesterov(learningRate, momentum))
    .list()
    .layer(0, new ConvolutionLayer.Builder(5,5)
    .nIn(1).nOut(100).activation(IDENTITY).build())
    // ... more layers here
    .layer(5, new OutputLayer.Builder()
        .lossFunction(NEGATIVELOGLIKELIHOOD).nOut(26)
        .activation(SOFTMAX).build())
    .backprop(true)
    .build();
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK



```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(132)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .weightInit(XAVIER)
    .updater(new Nesterov(learningRate, momentum))
    .list()
    .layer(0, new ConvolutionLayer.Builder(5,5)
    .nIn(1).nOut(100).activation(IDENTITY).build())
    // ... more layers here
    .layer(5, new OutputLayer.Builder()
        .lossFunction(NEGATIVELOGLIKELIHOOD).nOut(26)
        .activation(SOFTMAX).build())
    .backprop(true)
    .build();
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK



```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(132)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .weightInit(XAVIER)
    .updater(new Nesterov(learningRate, momentum))
    .list()
    .layer(0, new ConvolutionLayer.Builder(5,5)
    .nIn(1).nOut(100).activation(IDENTITY).build())
    // ... more layers here
    .layer(5, new OutputLayer.Builder()
        .lossFunction(NEGATIVELOGLIKELIHOOD).nOut(26)
        .activation(SOFTMAX).build())
    .backprop(true)
    .build();
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(132)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .weightInit(XAVIER)
    .updater(new Nesterov(learningRate, momentum))
    .list()
    .layer(0, new ConvolutionLayer.Builder(5,5)
    .nIn(1).nOut(100).activation(IDENTITY).build())
    // ... more layers here
    .layer(5, new OutputLayer.Builder()
        .lossFunction(NEGATIVELOGLIKELIHOOD).nOut(26)
        .activation(SOFTMAX).build())
    .backprop(true)
    .build();
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(132)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .weightInit(XAVIER)
    .updater(new Nesterov(learningRate, momentum))
    .list()
    .layer(0, new ConvolutionLayer.Builder(5,5)
    .nIn(1).nOut(100).activation(IDENTITY).build())
    // ... more layers here
    .layer(5, new OutputLayer.Builder()
        .lossFunction(NEGATIVELOGLIKELIHOOD).nOut(26)
        .activation(SOFTMAX).build())
    .backprop(true)
    .build();
```

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK

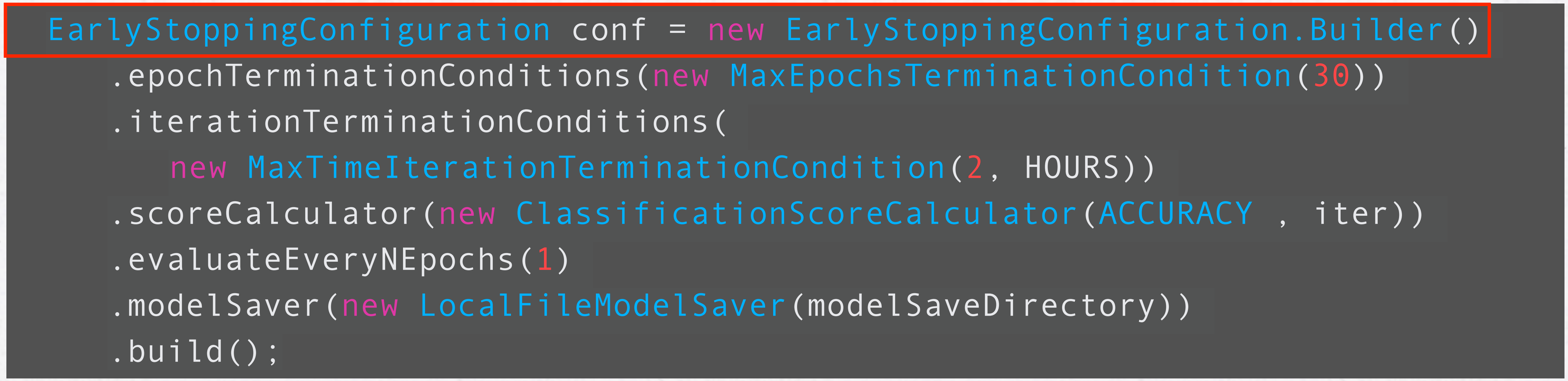
```
MultiLayerNetwork model = new MultilayerNetwork(conf);
```

This is the network configuration from last slide…

HOW TO BUILD: CONVOLUTIONAL NEURAL NETWORK

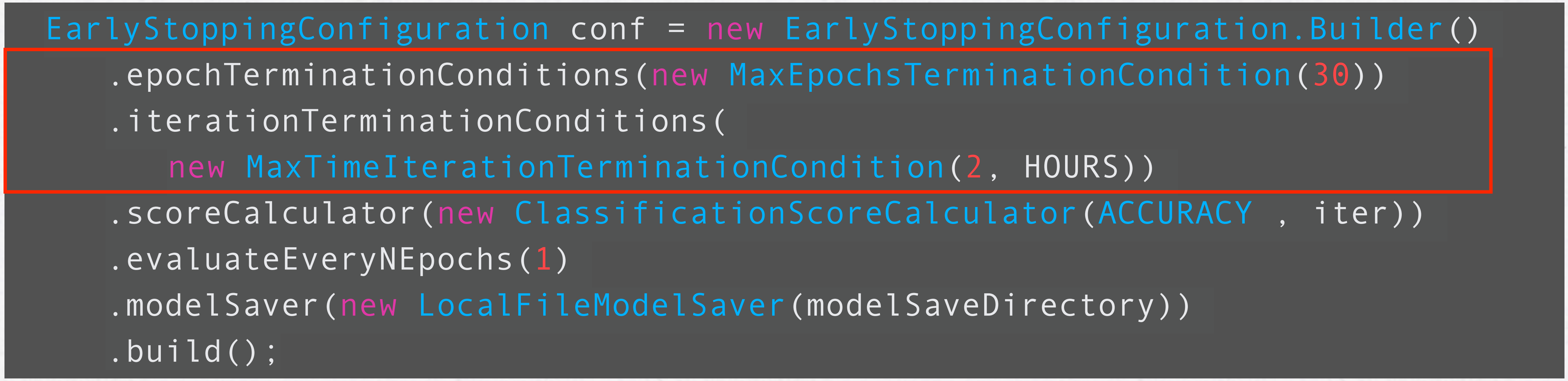
- We will build a convolutional neural network using Deeplearning4J
- Use Deeplearning4J MultilayerConfiguration to build a MultiLayerNetwork
- Use Deeplearning4J EarlyStoppingTrainer to train and save the network

HOW TO BUILD: EARLYSTOPPINGTRAINER



```java
EarlyStoppingConfiguration conf = new EarlyStoppingConfiguration.Builder()
    .epochTerminationConditions(new MaxEpochsTerminationCondition(30))
    .iterationTerminationConditions(
        new MaxTimeIterationTerminationCondition(2, HOURS))
    .scoreCalculator(new ClassificationScoreCalculator(ACCURACY , iter))
    .evaluateEveryNEpochs(1)
    .modelSaver(new LocalFileModelSaver(modelSaveDirectory))
    .build();
```

HOW TO BUILD: EARLYSTOPPINGTRAINER



```
EarlyStoppingConfiguration conf = new EarlyStoppingConfiguration.Builder()
   .epochTerminationConditions(new MaxEpochsTerminationCondition(30))
   .iterationTerminationConditions(
      new MaxTimeIterationTerminationCondition(2, HOURS))
   .scoreCalculator(new ClassificationScoreCalculator(ACCURACY , iter))
   .evaluateEveryNEpochs(1)
   .modelSaver(new LocalFileModelSaver(modelSaveDirectory))
   .build();
```

HOW TO BUILD: EARLYSTOPPINGTRAINER

```
EarlyStoppingConfiguration conf = new EarlyStoppingConfiguration.Builder()
    .epochTerminationConditions(new MaxEpochsTerminationCondition(30))
    .iterationTerminationConditions(
        new MaxTimeIterationTerminationCondition(2, HOURS))
    .scoreCalculator(new ClassificationScoreCalculator(ACCURACY , iter))
    .evaluateEveryNEpochs(1)
    .modelSaver(new LocalFileModelSaver(modelSaveDirectory))
    .build();
```

HOW TO BUILD: EARLYSTOPPINGTRAINER

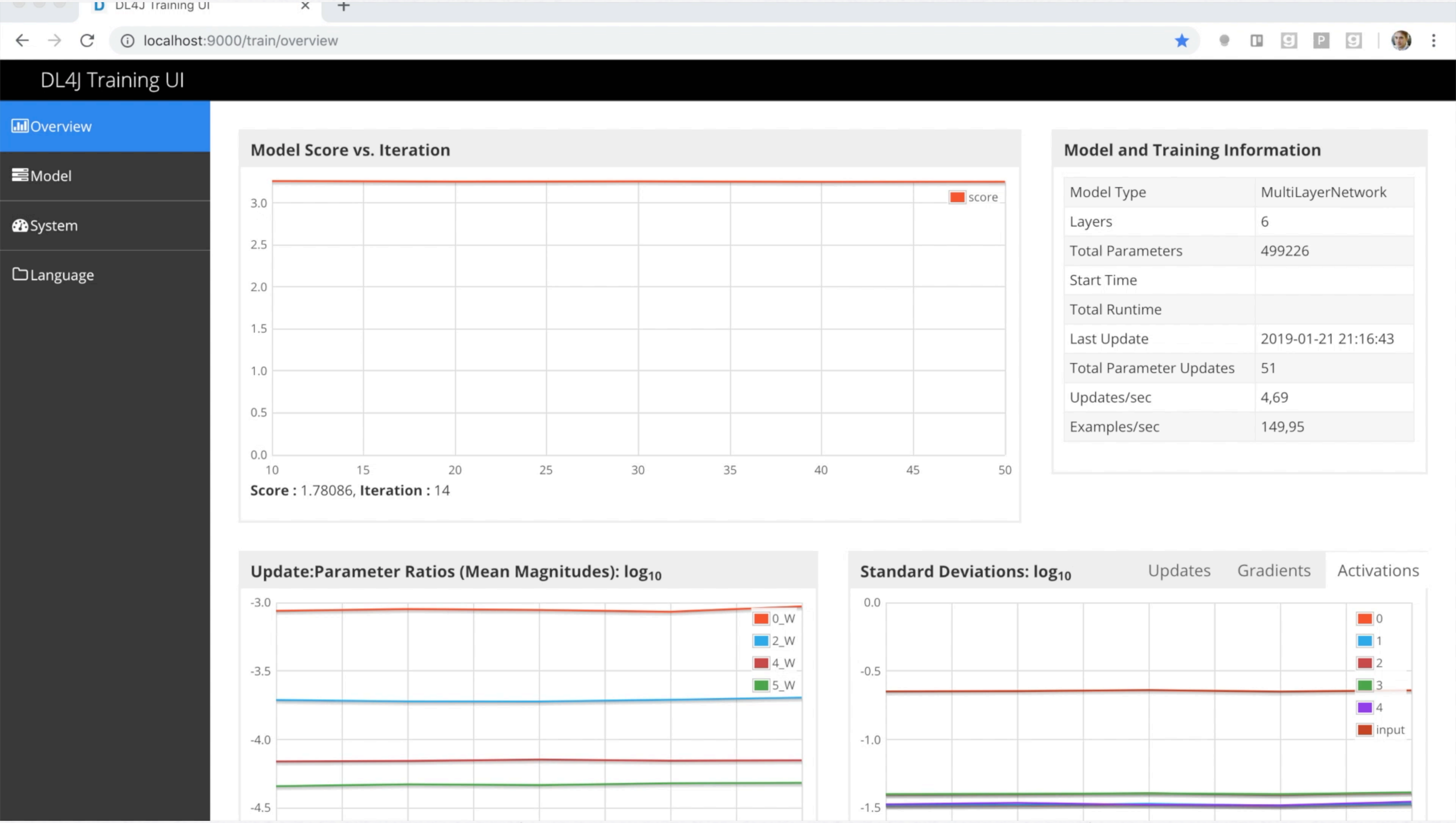
```
EarlyStoppingConfiguration conf = new EarlyStoppingConfiguration.Builder()
   .epochTerminationConditions(new MaxEpochsTerminationCondition(30))
   .iterationTerminationConditions(
      new MaxTimeIterationTerminationCondition(2, HOURS))
   .scoreCalculator(new ClassificationScoreCalculator(ACCURACY , iter))
   .evaluateEveryNEpochs(1)
   .modelSaver(new LocalFileModelSaver(modelSaveDirectory))
   .build();
```

HOW TO BUILD: EARLYSTOPPINGTRAINER

```java
EarlyStoppingTrainer trainer = new EarlyStoppingTrainer(conf, model,
datasetIterator);
```

MovieTime

fresh
POP CORN
DELICIOUS
NUTRITIOUS
fresh
POP CORN
DELICIOUS
CRISP



Model Score vs. Iteration

Score : 1.78086, **Iteration :** 14

Model and Training Information

Model Type	MultiLayerNetwork
Layers	6
Total Parameters	499226
Start Time	
Total Runtime	
Last Update	2019-01-21 21:16:43
Total Parameter Updates	51
Updates/sec	4,69
Examples/sec	149,95

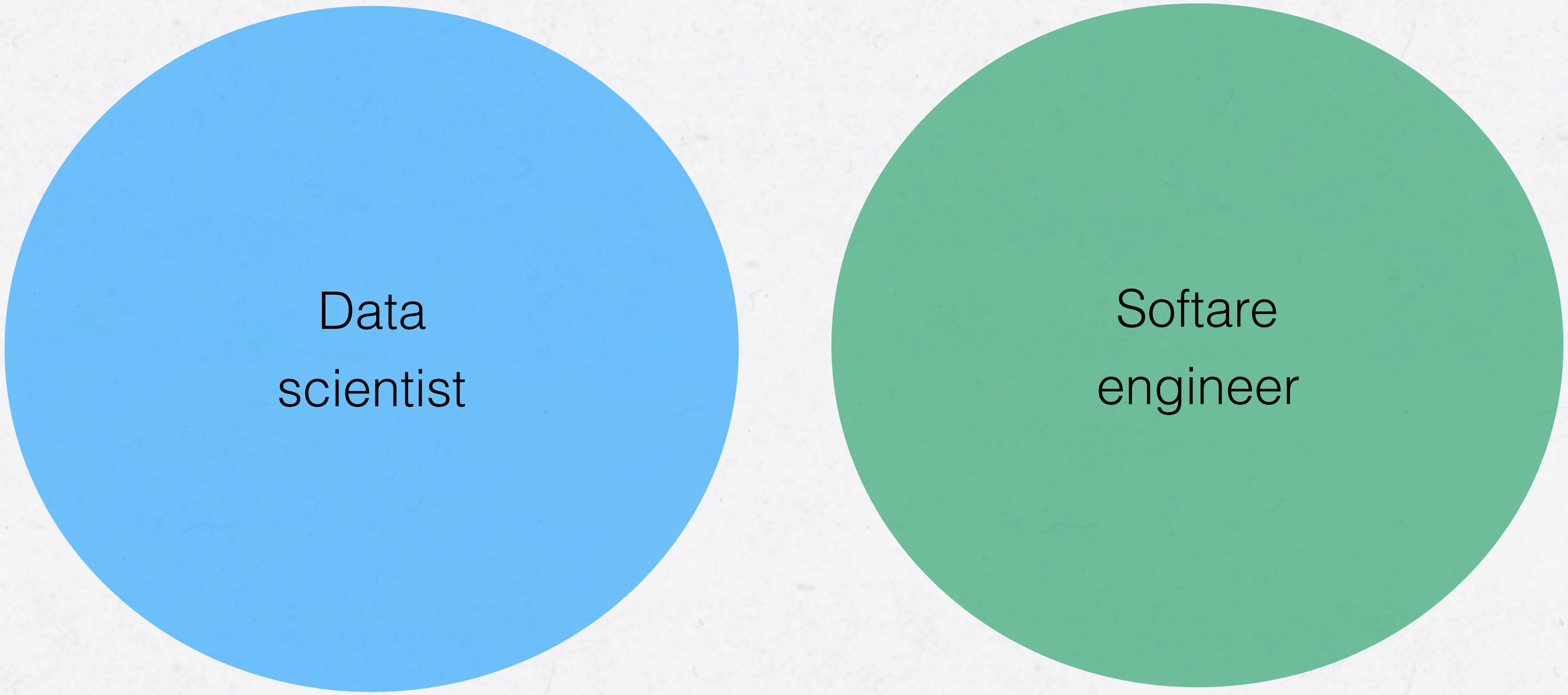
Update:Parameter Ratios (Mean Magnitudes): $\log_{10}$

Standard Deviations: $\log_{10}$

THE RESULT

DEMO!

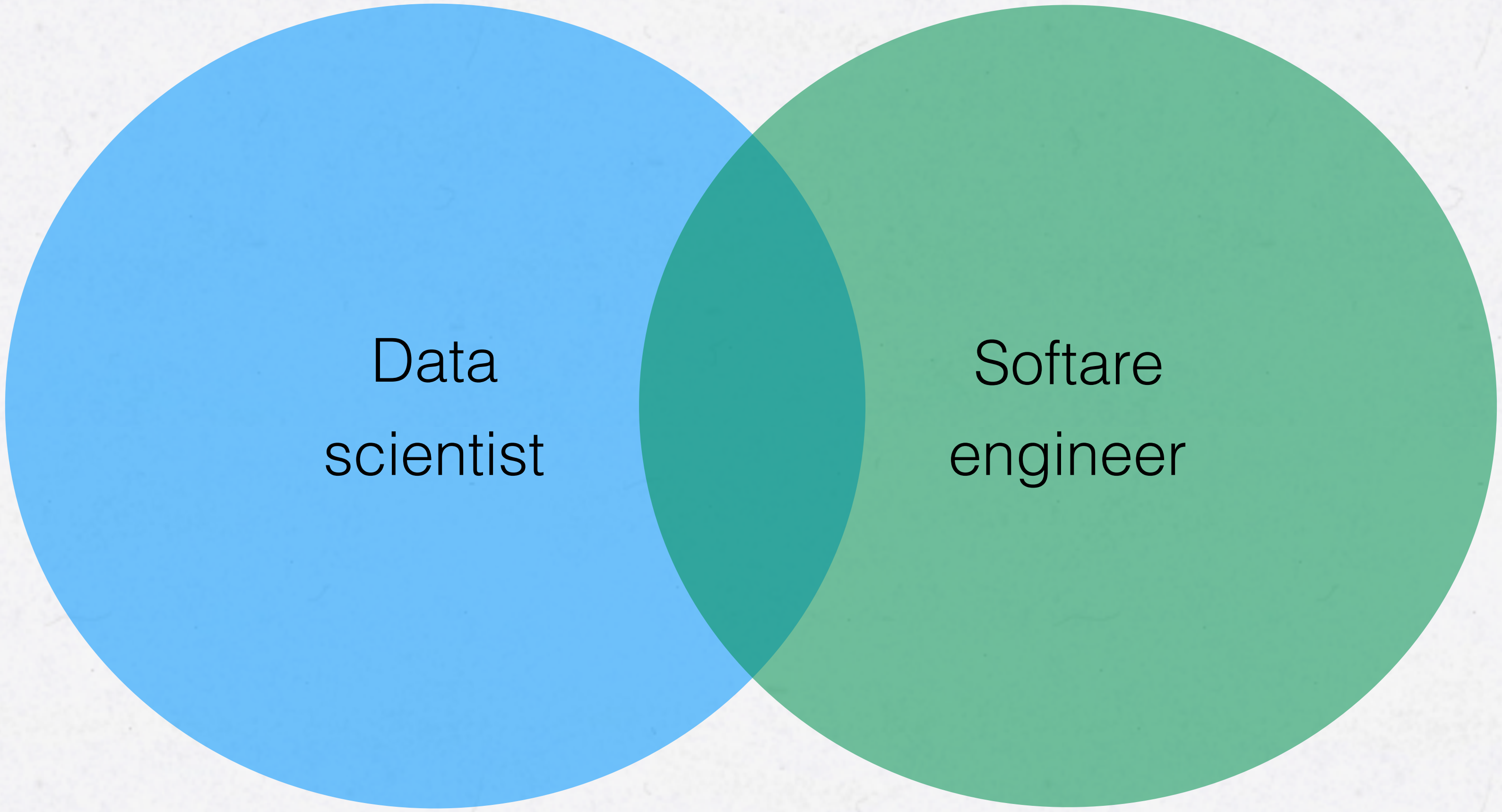
LOOKING AHEAD: IS THIS SOMETHING FOR ME?



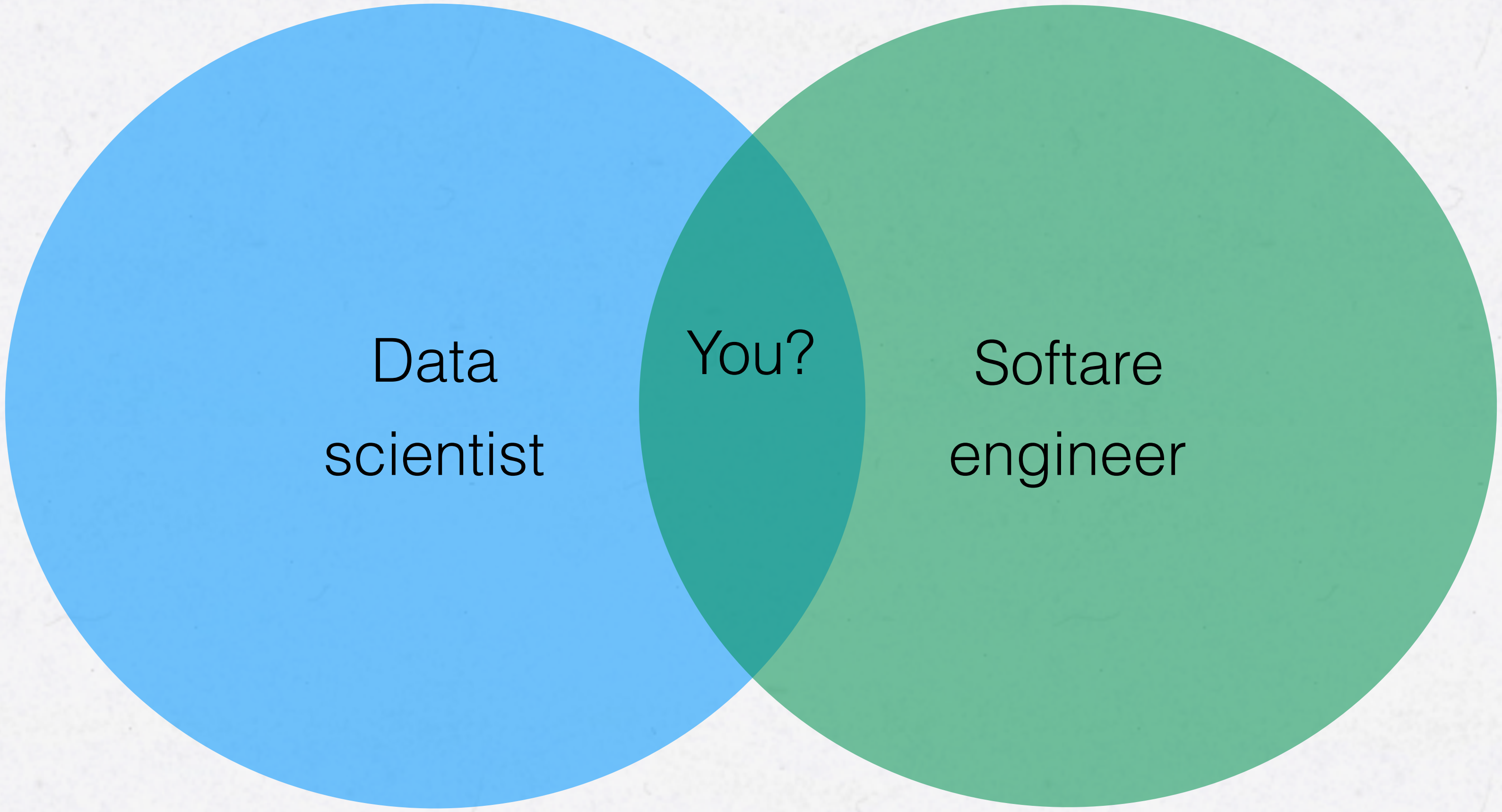
Data scientist

Softare engineer

LOOKING AHEAD: IS THIS SOMETHING FOR ME?



LOOKING AHEAD: IS THIS SOMETHING FOR ME?



LOOKING AHEAD

- Possibilities and risks, how do we measure them?
- Ethics and responsibilities?
- 5 years ahead vs 30 years ahead?

LOOKING AHEAD: POSSIBILITIES

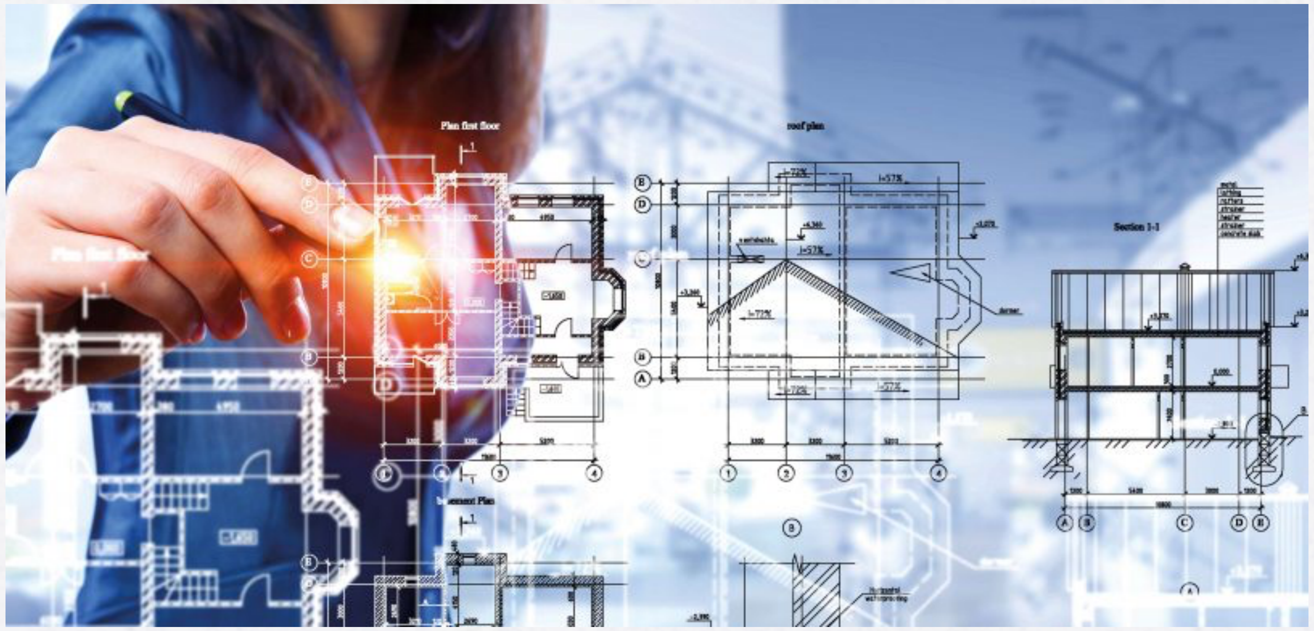
- If we can have self-driving cars, what else can we have?

LOOKING AHEAD: POSSIBILITIES

- If we can have self-driving cars, what else can we have?
  - Self-cooking kitchens?

LOOKING AHEAD: POSSIBILITIES

- If we can have self-driving cars, what else can we have?
  - Self-cooking kitchens?
  - Self-building buildings?

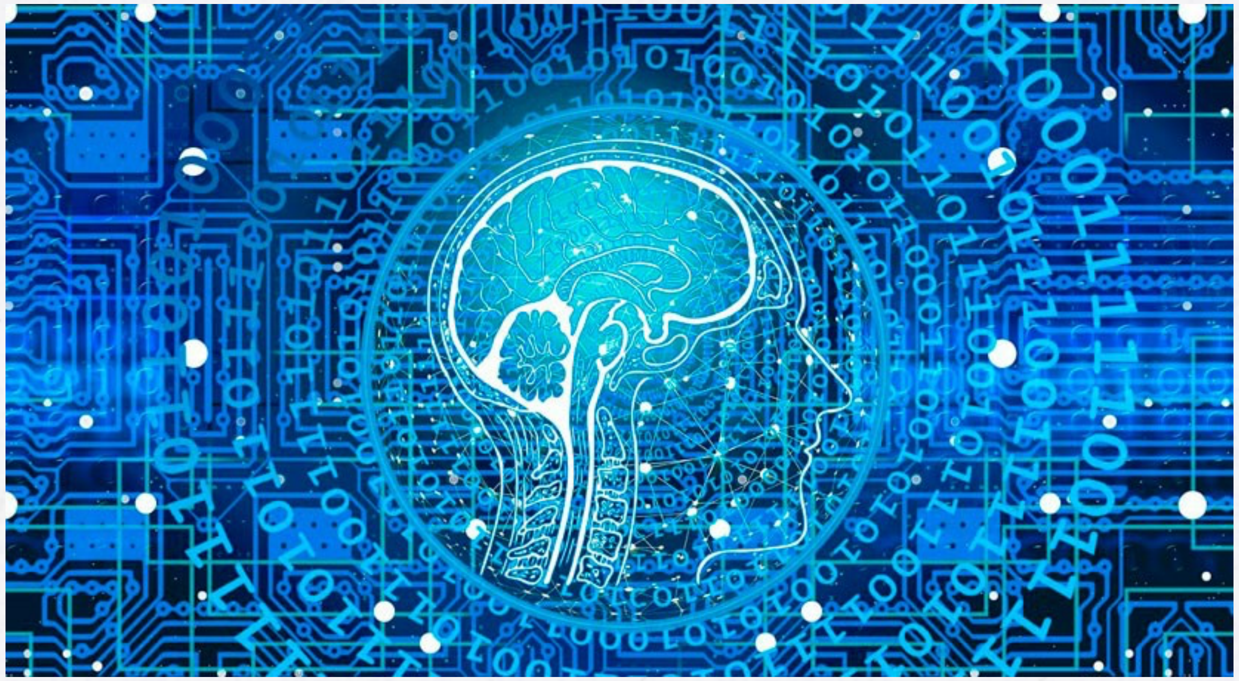


LOOKING AHEAD: POSSIBILITIES

- If we can have self-driving cars, what else can we have?
  - Self-cooking kitchens?
  - Self-building buildings?
  - Smarter production machines that can be used for almost all production

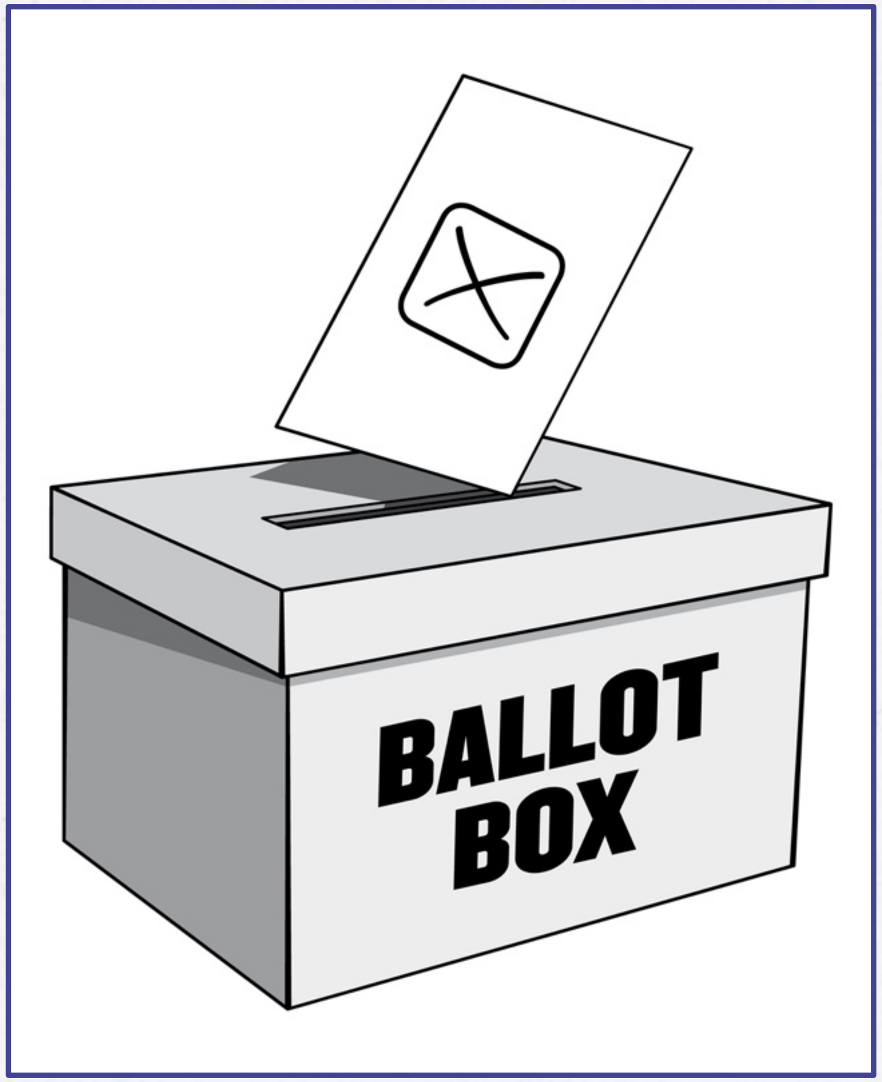
LOOKING AHEAD: POSSIBILITIES

- If we can have self-driving cars, what else can we have?
  - Self-cooking kitchens?
  - Self-building buildings?
  - Smarter production machines that can be used for almost all production
  - Massively improved forecasts eliminating waste, optimised planning
  - Etc.



LOOKING AHEAD: RISKS

- Influencing democratic votes?
  - Cambridge Analytica, fake news, etc.



LOOKING AHEAD: RISKS

- Influencing democratic votes?
  - Cambridge Analytica, fake news, etc.
- Job loss?
  - *"Robot automation will 'take 800 million jobs by 2030'"* (McKinsey Global Institute)

LOOKING AHEAD: RISKS

- Influencing democratic votes?
  - Cambridge Analytica, fake news, etc.
- Job loss?
  - *"Robot automation will 'take 800 million jobs by 2030'"* (McKinsey Global Institute)
- Surveillance society?

LOOKING AHEAD: RISKS

- Influencing democratic votes?
  - Cambridge Analytica, fake news, etc.
- Job loss?
  - *"Robot automation will 'take 800 million jobs by 2030'"* (McKinsey Global Institute)
- Surveillance society?
- What risks do you see?

SUMMARY

- Machine learning is about utilizing patterns in data, to make classifications, forecasts, or grouping of data
- There are several different frameworks available for machine learning in Java. Some of them are focused on traditional machine learning algorithms.
- Deeplearning4J is a Java framework for developing machine learning systems using deep learning algorithms such as artificial neural networks.
- With modern frameworks like Deeplearning4J in Java or Keras in Python, much of the complexity of machine learning algorithms are abstracted away.
- Machine learning is about to impose mayor changes to all parts of our society, for good or bad. Are you prepared?

Thank You!

This session was for You!
I hope You will walk away from this with some new ideas and insights

I hope You have enjoyed it!