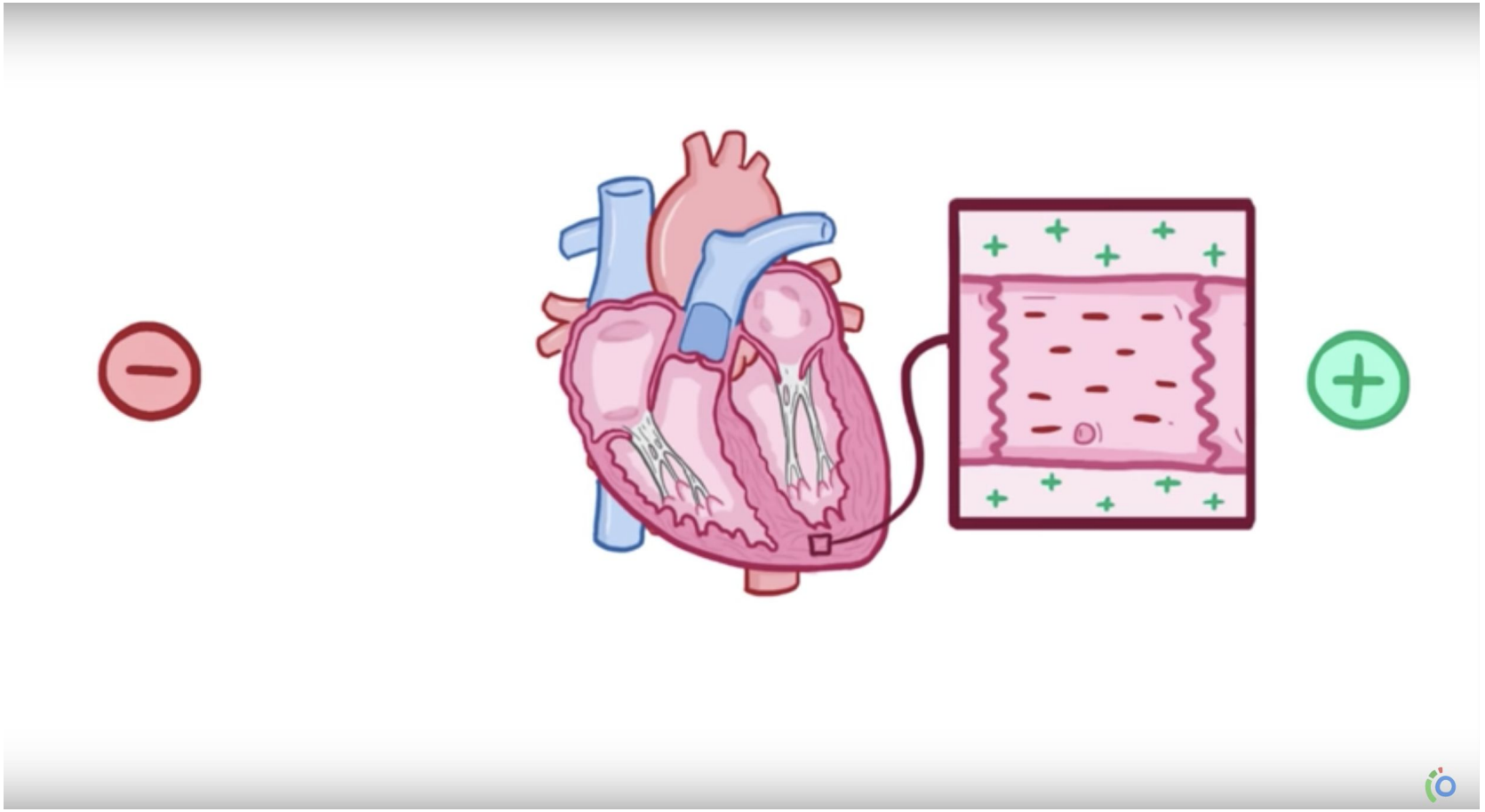
ML4HC Project 3: ECG time series

ECG

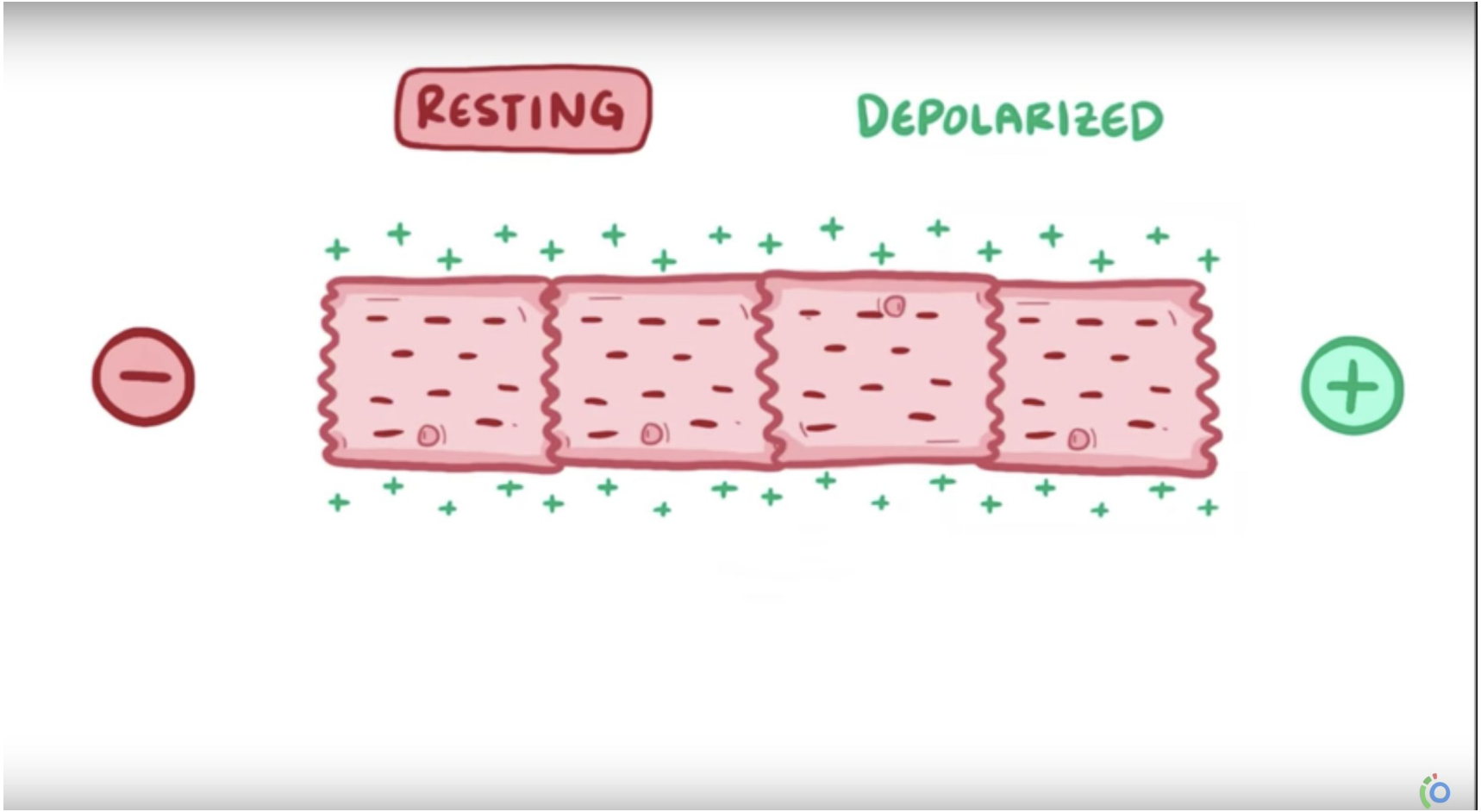
- Electrocardiography is the process of producing an electrocardiogram (ECG or EKG), a recording - a graph of voltage versus time - of the electrical activity of the heart using electrodes placed on the skin.
- Electrodes detect the small electrical changes that are a consequence of cardiac muscle depolarization followed by repolarization during each cardiac cycle (heartbeat).
- In biology, depolarization is a change within a cell, during which the cell undergoes a shift in electric charge distribution, resulting in less negative charge inside the cell.
- Changes in the normal ECG pattern occur in numerous cardiac abnormalities, including cardiac rhythm disturbances, inadequate coronary artery blood flow, etc.

EEG channels / leads

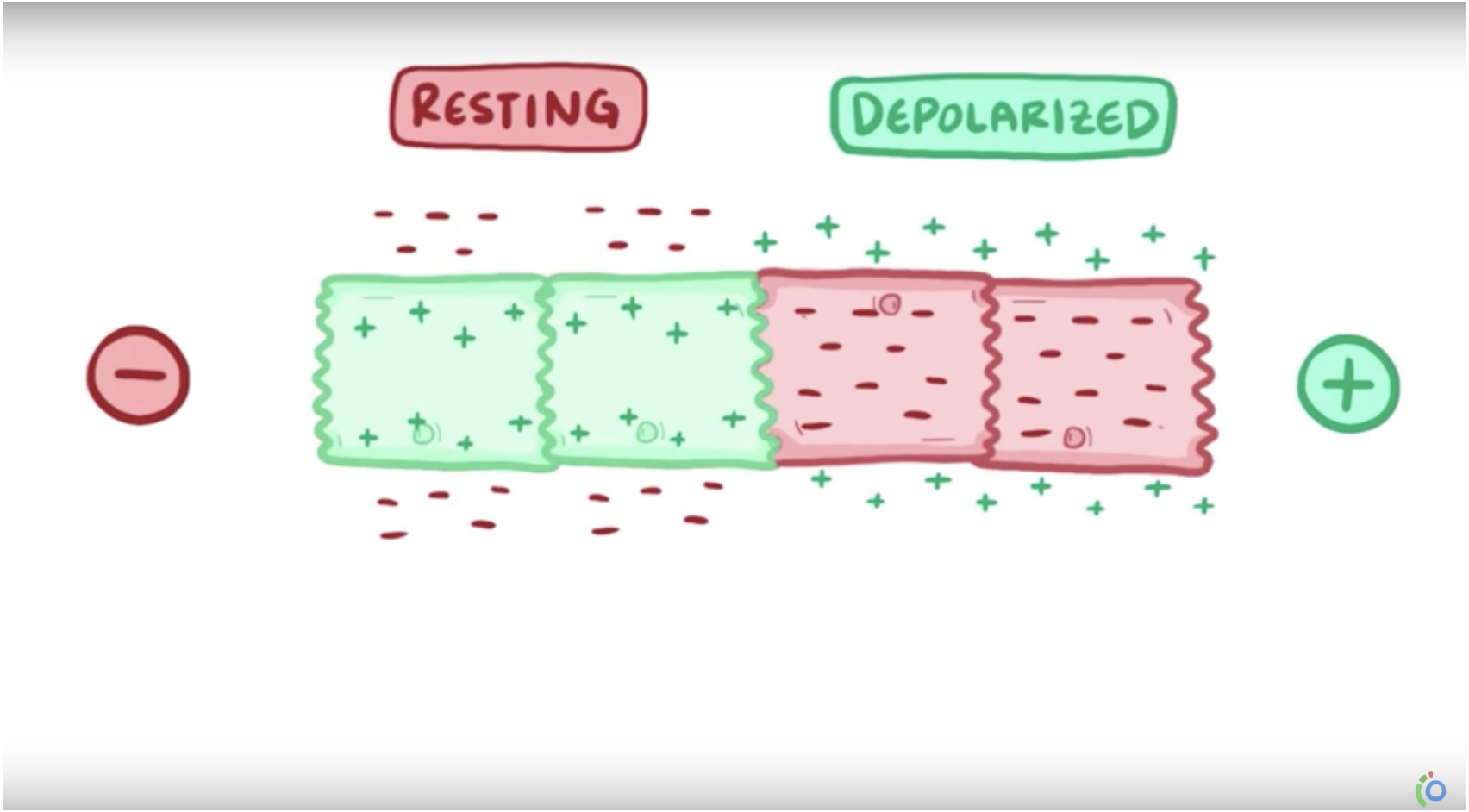
- In a conventional 12-lead ECG, 10 electrodes are placed on the patient's limbs and on the surface of the chest.
- The overall magnitude of the heart's electrical potential is then measured from 12 different angles ("leads") and is recorded over a period of time (usually ten seconds).
- Electrodes are placed in standardized locations.



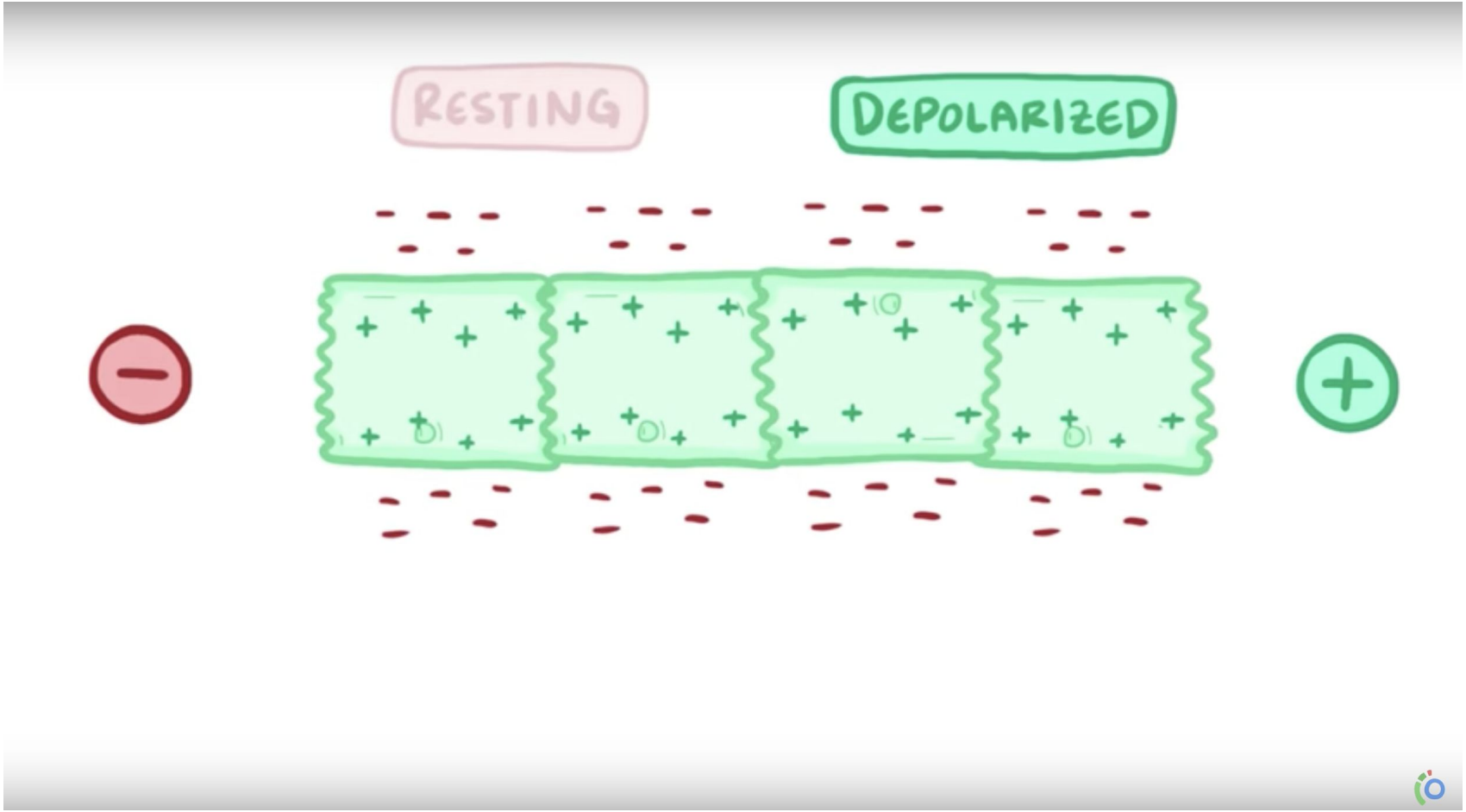
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



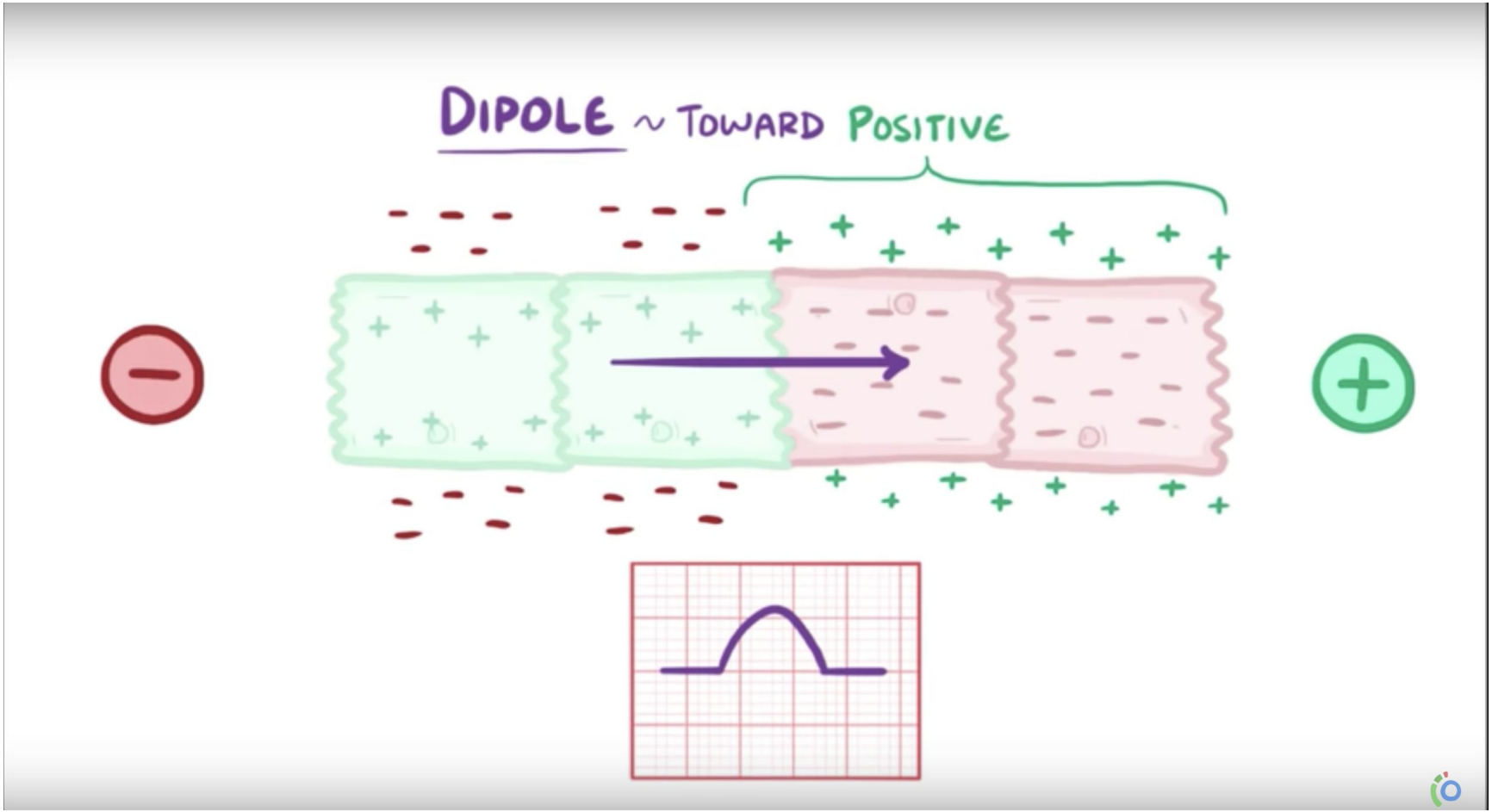
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



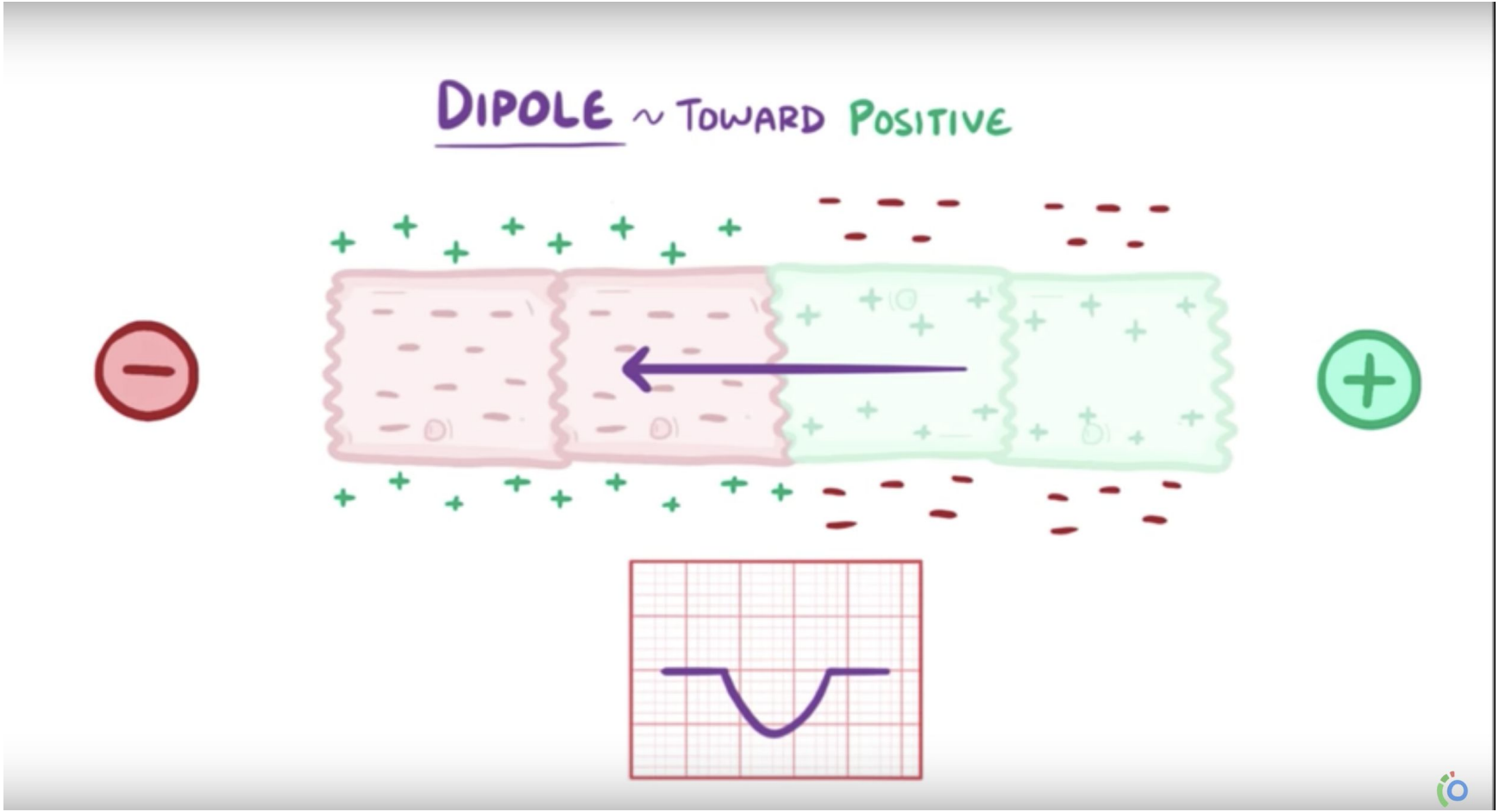
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



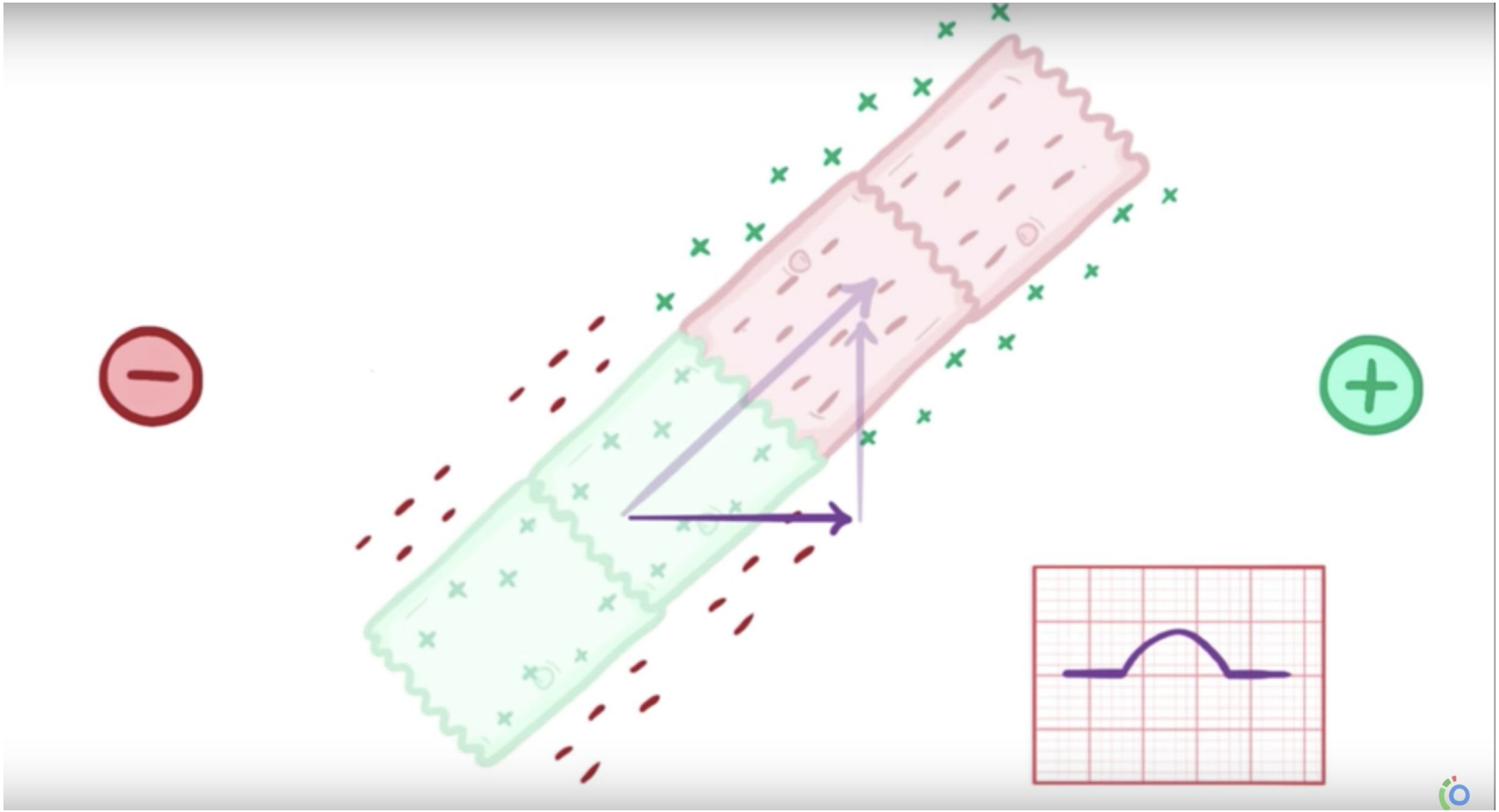
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



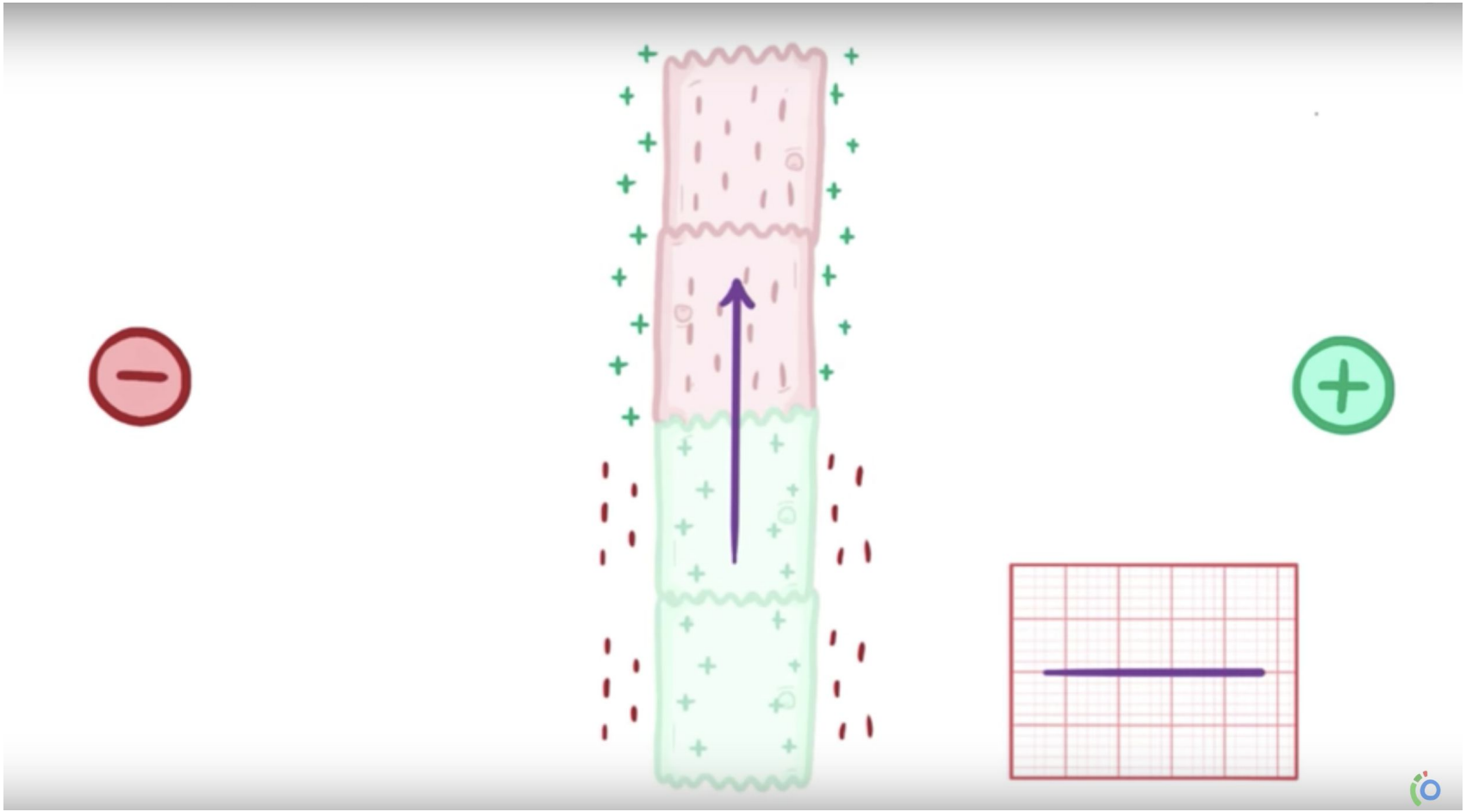
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



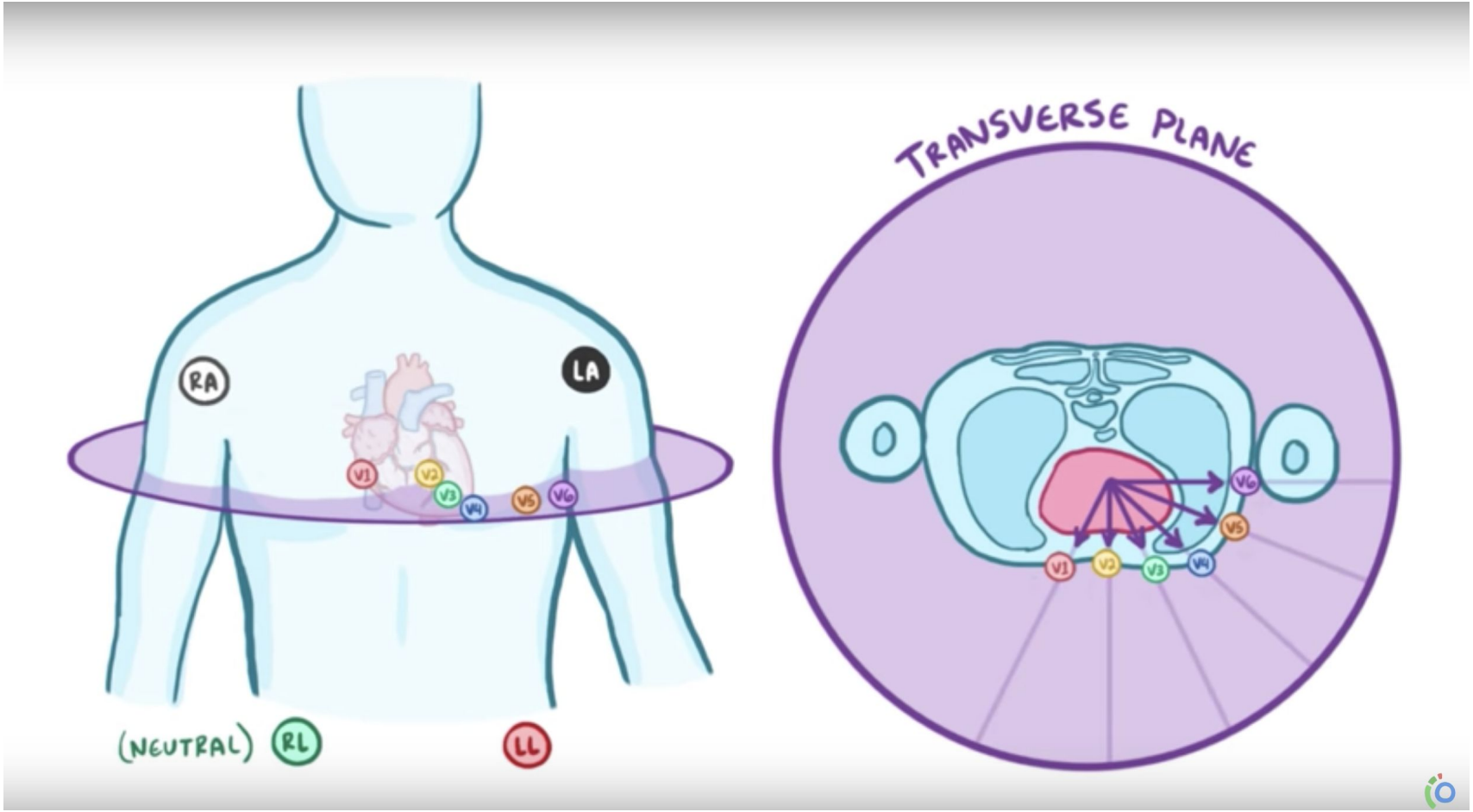
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



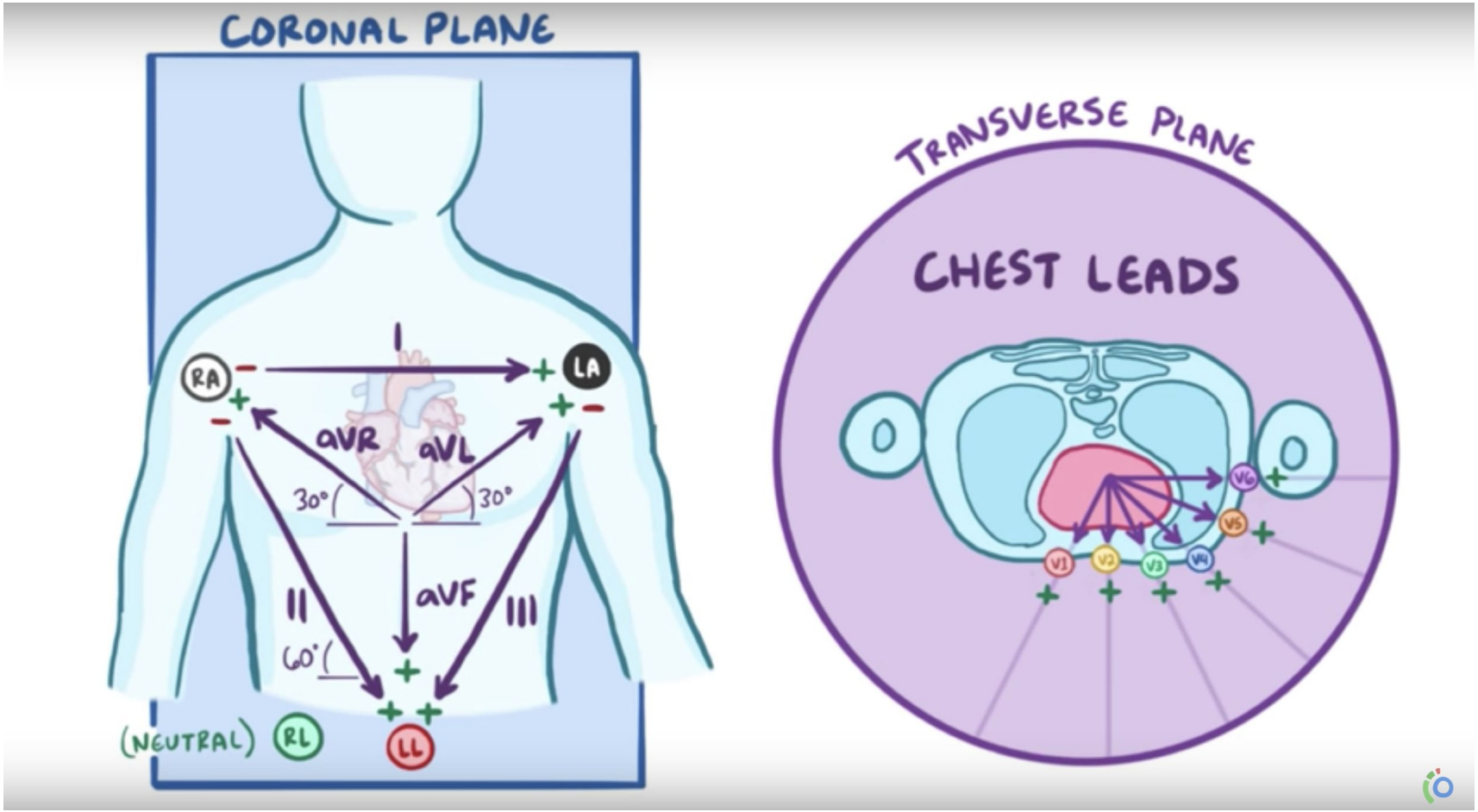
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



* from https://www.youtube.com/watch?v=xIZQRjkwV9Q

Apple watch ECG

- 1 channel ECG.
- “The ability of the ECG app to accurately classify an ECG recording into AFib (atrial fibrillation) and sinus rhythm was tested in a clinical trial of approximately 600 subjects, and demonstrated 99.6% specificity with respect to sinus rhythm classification and 98.3% sensitivity for AFib classification for the classifiable results.”
- “AFib is a type of irregular heart rhythm. AFib occurs when the heart beats in an irregular pattern. It’s a common form of irregular heart rhythm where the upper chambers of the heart beat out of sync with the lower chambers. “
- “Approximately 2% of people younger than 65 years old and 9% of people 65 and older have AFib. Irregularities in heart rhythm become more common as people get older”
- “Some individuals with AFib don’t experience any symptoms.”
- “If left untreated, AFib can lead to heart failure or blood clots that may lead to stroke”

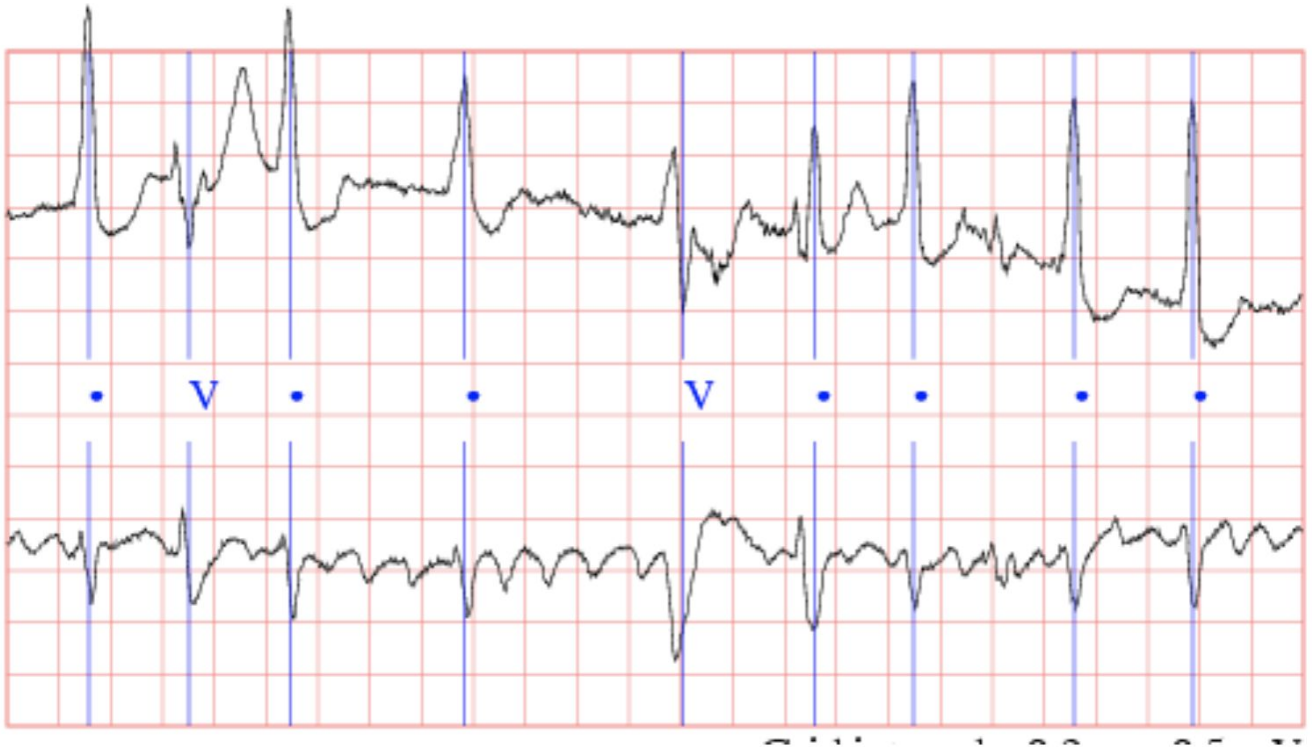
Motivation

- Personal devices that do ECG will become more and more frequent.
- ML systems will be needed to automatically evaluate all this data.

MIT-BIH Arrhythmia Database

- https://physionet.org/physiobank/database/mitdb/
- 47 subjects.
- 48 half-hour excerpts of two-channel ambulatory ECG recordings.
- The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range.
- Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database.

MIT-BIH Arrhythmia Database



The PTB Diagnostic ECG Database

- https://physionet.org/physiobank/database/ptbdb/
- 16 input channels, (14 for ECGs, 1 for respiration, 1 for line voltage)
- Input voltage: ±16 mV, compensated offset voltage up to ± 300 mV
- Input resistance: 100 Ω (DC)
- Resolution: 16 bit with 0.5 μV/LSB (2000 A/D units per mV)
- Bandwidth: 0 - 1 kHz (synchronous sampling of all channels)
- Noise voltage: max. 10 μV (pp), respectively 3 μV (RMS) with input short circuit
- Online recording of skin resistance
- Noise level recording during signal collection

The PTB Diagnostic ECG Database

- 549 records from 290 subjects (aged 17 to 87, mean 57.2; 209 men, mean age 55.5, and 81 women, mean age 61.6; ages were not recorded for 1 female and 14 male subjects).
- Each subject is represented by one to five records.
- Each record includes 15 simultaneously measured signals: the conventional 12 leads (i, ii, iii, avr, avl, avf, v1, v2, v3, v4, v5, v6) together with the 3 Frank lead ECGs (vx, vy, vz).
- Each signal is digitized at 1000 samples per second, with 16 bit resolution over a range of ± 16.384 mV.
- On special request to the contributors of the database, recordings may be available at sampling rates up to 10 KHz.
- Within the header (.hea) file of most of these ECG records is a detailed clinical summary, including age, gender, diagnosis, and where applicable, data on medical history, medication and interventions, coronary artery pathology, ventriculography, echocardiography, and hemodynamics.

The PTB Diagnostic ECG Database

Diagnostic class	Number of subjects
Myocardial infarction	148
Cardiomyopathy/Heart failure	18
Bundle branch block	15
Dysrhythmia	14
Myocardial hypertrophy	7
Valvular heart disease	6
Myocarditis	4
Miscellaneous	4
Healthy controls	52

The PTB Diagnostic ECG Database

ECG Heartbeat Classification: A Deep Transferable Representation

Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh
University of California, Los Angeles (UCLA)
Los Angeles, USA

ECG Heartbeat Classification: A Deep Transferable Representation

- Authors propose a novel framework for ECG analysis that is able to represent the signal in a way that is transferable between different tasks.
- Deep neural network architecture
- Show that the signal representation learned from one task is successfully transferable to other task using ECG signals.

ECG Heartbeat Classification: A Deep Transferable Representation
Data preprocessing

- In all the experiments, authors have used ECG lead II re-sampled to the sampling frequency of 125Hz as the input.
- From MIT-BIH Arrhythmia Database, authors use annotations in this dataset to create five different beat categories in accordance with Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard.
- The PTB Diagnostics dataset consists of ECG records from 290 subjects: 148 diagnosed as MI, 52 healthy control, and the rest are diagnosed with 7 different disease. In this study authors used ECG worked with MI and healthy control categories in our analyses (binary classification task).
- Store each heartbeat as an independent sample (tricky for the evaluation…).
- Samples are downsampled, normalized and padded.
- Sample labels are in the last column!

ECG Heartbeat Classification: A Deep Transferable Representation

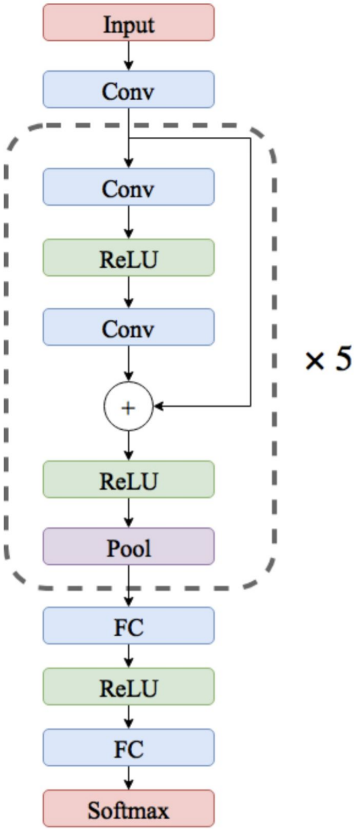


Fig. 2: Architecture of the proposed network.

ECG Heartbeat Classification: A Deep Transferable Representation

TABLE II: Comparison of heartbeat classification results.

Work	Approach	Average Accuracy (%)
This Paper	**Deep residual CNN**	**93.4**
Acharya *et al.* [23]	Augmentation + CNN	93.5
Martis *et al.* [24]	DWT + SVM	93.8
Li *et al.* [25]	DWT + random forest	94.6

TABLE III: Comparison of MI classification results.

Work	Accuracy (%)	Precision (%)	Recall (%)
This Paper[1]	**95.9**	**95.2**	**95.1**
Acharya *et al.* [27][1]	93.5	92.8	93.7
Safdarian *et al.* [28][1]	94.7	–	–
Kojuri *et al.* [29][2]	95.6	97.9	93.3
Sun *et al.* [30][3]	–	82.4	92.6
Liu *et al.* [31][3]	94.4	–	–
Sharma *et al.* [26][3]	96	99	93

[1]: PTB dataset, ECG lead II
[2]: dataset collected by authors, 12-lead ECG
[3]: PTB dataset, 12-lead ECG

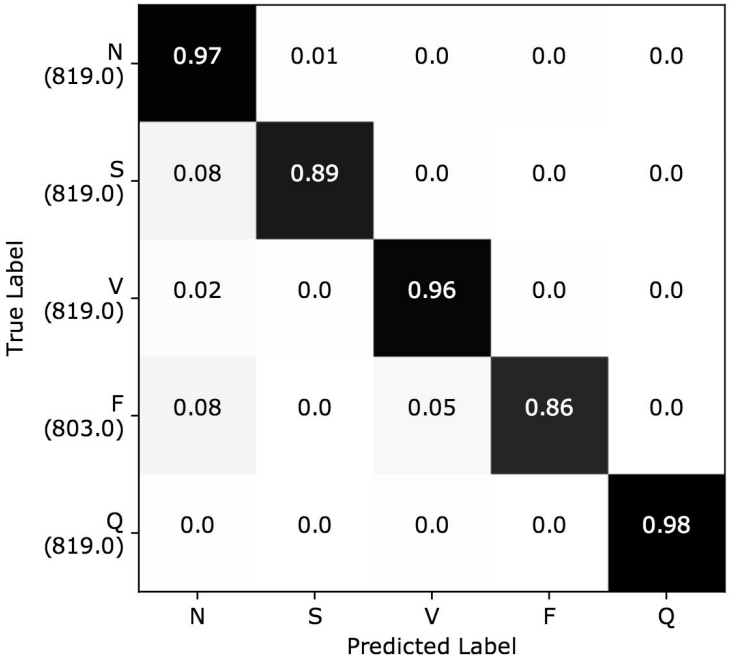


Fig. 3: Confusion matrix for heartbeat classification on the test set. Total number of samples in each class is indicated inside parenthesis. Numbers inside blocks are number of samples classified in each category normalized by the total number of samples and rounded to two digits.

ECG Heartbeat Classification: A Deep Transferable Representation
Baseline model

- Shared dataset is not equal to the one of the paper.
- Use as baselines:
  - https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_mitbih.py
  - https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_ptbdb.py
- Unlike proposed model, this implementation doesn’t contain residual blocks.
- Don’t take the transfer learning code of that repository.

Tasks: solve both datasets independently

- Samples visualization and clustering.
- Solve both datasets with RNNs.
  - For the binary one, report accuracy, AUROC and AUPRC.
  - For the non-binary one, report accuracy.
- Compare with other models (standard models, bidirectional RNN, CNN with residual blocks, etc).
- Ensemble of models (e.g. average of the outputs, logistic regression on the outputs, etc).

Tasks (optional, but at least everyone should try): transfer learning

- Transfer learning with RNNs, frozen base model.
  - Train RNN model with MIT-BIH Arrhythmia Database.
  - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model) and feed it with the PTB Diagnostic ECG Database. The outputs are the vector representation of the PTB Diagnostic ECG Database samples.
  - Take the representations of the samples of PTB Diagnostic ECG Database and train a new small feedforward neural network equivalent to the layers you removed from the first model.

- Transfer learning with RNNs, re-training whole model.
  - Train RNN model with MIT-BIH Arrhythmia Database.
  - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model)
  - Add output layer(s) for the PTB Diagnostic ECG Database and train the whole model.

Tasks (optional, but at least everyone should try): transfer learning

- Transfer learning with RNNs, first frozen base model, then re-training whole model.
  - Train RNN model with MIT-BIH Arrhythmia Database.
  - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model).
  - Freeze the layers of the model (https://keras.io/getting-started/faq/#how-can-i-freeze-keras-layers)
  - Add output layer(s) for the PTB Diagnostic ECG Database and train the output layers.
  - Unfreeze the whole model.
  - Train the whole model.

What you are given

- Datasets (download from kaggle).
- Baseline code, containing both baseline results and data splits you have to use.
- Fixed train / test splits for the dataset.
  - MIT-BIH Arrhythmia Database: code to compute the split (specific random seed).
  - PTB Diagnostic ECG Database: two separate files.

Example of RNN with Keras

Sequence classification with LSTM:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import Embedding
from keras.layers import LSTM

max_features = 1024

model = Sequential()
model.add(Embedding(max_features, output_dim=256))
model.add(LSTM(128))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=16, epochs=10)
score = model.evaluate(x_test, y_test, batch_size=16)
```

Keras functional API

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels)  # starts training
```

Hyperparameter search

Divide in train/validation/set.

Find the best hyperparameters for the validation set:

- Input layer size.
- Recurrent layer hidden state size.
- Dropout rate.
- Number of layers.
- Optimizer.
- Learning rate.
- Recurrent gate (standard RNN, LSTM, GRU).

Masking [source]

```
keras.layers.Masking(mask_value=0.0)
```

Masks a sequence by using a mask value to skip timesteps.

If all features for a given sample timestep are equal to `mask_value`, then the sample timestep will be masked (skipped) in all downstream layers (as long as they support masking).

If any downstream layer does not support masking yet receives such an input mask, an exception will be raised.

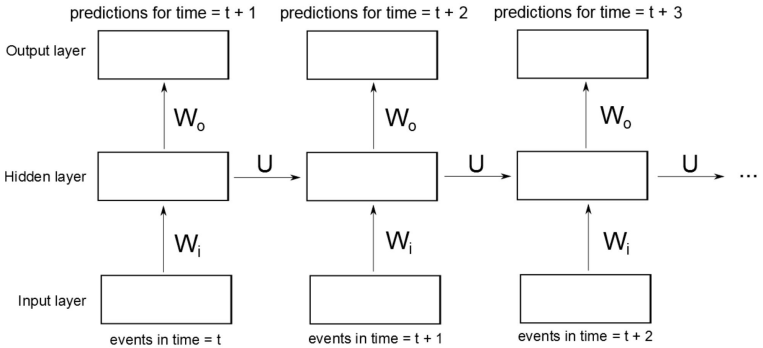
Example

Consider a Numpy data array `x` of shape `(samples, timesteps, features)`, to be fed to an LSTM layer. You want to mask sample #0 at timestep #3, and sample #2 at timestep #5, because you lack features for these sample timesteps. You can do:

- set `x[0, 3, :] = 0.` and `x[2, 5, :] = 0.`
- insert a `Masking` layer with `mask_value=0.` before the LSTM layer:

```
model = Sequential()
model.add(Masking(mask_value=0., input_shape=(timesteps, features)))
model.add(LSTM(32))
```

RNNs can return one vector or a sequence



Docs » Layers » Recurrent Layers

RNN

```
keras.layers.RNN(cell, return_sequences=False, return_state=False, go_backwards
```

TimeDistributed [source]

```
keras.layers.TimeDistributed(layer)
```

This wrapper applies a layer to every temporal slice of an input.

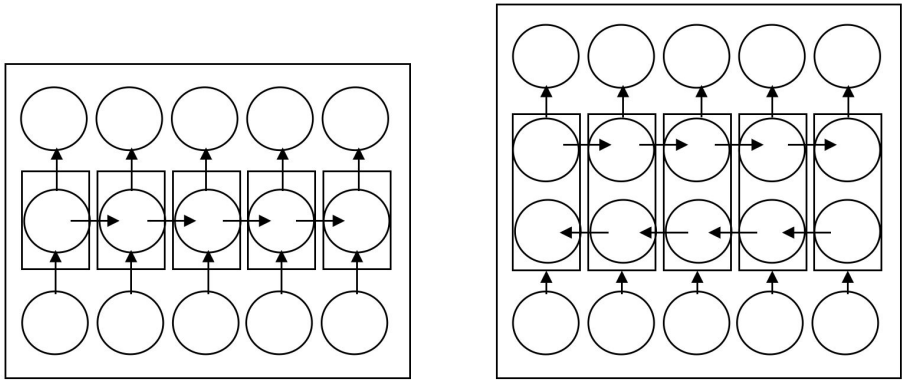
The input should be at least 3D, and the dimension of index one will be considered to be the temporal dimension.

Consider a batch of 32 samples, where each sample is a sequence of 10 vectors of 16 dimensions. The batch input shape of the layer is then `(32, 10, 16)`, and the `input_shape`, not including the samples dimension, is `(10, 16)`.

You can then use `TimeDistributed` to apply a `Dense` layer to each of the 10 timesteps, independently:

```
# as the first layer in a model
model = Sequential()
model.add(TimeDistributed(Dense(8), input_shape=(10, 16)))
# now model.output_shape == (None, 10, 8)
```

Bidirectional RNN



(a) (b)

Structure overview

(a) unidirectional RNN
(b) bidirectional RNN

Bidirectional

```
keras.layers.Bidirectional(layer, merge_mode='concat', weights=None)
```

Bidirectional wrapper for RNNs.

Arguments

- **layer**: `Recurrent` instance.
- **merge_mode**: Mode by which outputs of the forward and backward RNNs will be comb
  One of {'sum', 'mul', 'concat', 'ave', None}. If None, the outputs will not be combined, they
  returned as a list.
- **weights**: Initial weights to load in the Bidirectional model

Raises

- **ValueError**: In case of invalid `merge_mode` argument.

Examples

```
model = Sequential()
model.add(Bidirectional(LSTM(10, return_sequences=True),
                        input_shape=(5, 10)))
model.add(Bidirectional(LSTM(10)))
model.add(Dense(5))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='rmsprop')
```

Deliverables

- Conda environment.
- Tasks specified in the Task slides. Do mandatory and at least try the optional ones.
- You can use different jupyter notebooks for different models/tasks.
- Use one notebook to index / summarize / present the results.
- Do not hardcode any results!
- Again: sequential execution and reproducibility !!
- Deadline: 01.05.2019

Tasks: solve both datasets independently

- Samples visualization and clustering.
- Solve both datasets with RNNs.
  - For the binary one, report accuracy, AUROC and AUPRC.
  - For the non-binary one, report accuracy.
- Compare with other models (standard models, bidirectional RNN, CNN with residual blocks, etc).
- Ensemble of models (e.g. average of the outputs, logistic regression on the outputs, etc).

Tasks (optional, but at least everyone should try): transfer learning

- Transfer learning with RNNs, frozen base model.
  - Train RNN model with MIT-BIH Arrhythmia Database.
  - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model) and feed it with the PTB Diagnostic ECG Database. The outputs are the vector representation of the PTB Diagnostic ECG Database samples.
  - Take the representations of the samples of PTB Diagnostic ECG Database and train a new small feedforward neural network equivalent to the layers you removed from the first model.

- Transfer learning with RNNs, re-training whole model.
  - Train RNN model with MIT-BIH Arrhythmia Database.
  - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model)
  - Add output layer(s) for the PTB Diagnostic ECG Database and train the whole model.

Tasks (optional, but at least everyone should try): transfer learning

- Transfer learning with RNNs, first frozen base model, then re-training whole model.
  - Train RNN model with MIT-BIH Arrhythmia Database.
  - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model).
  - Freeze the layers of the model (https://keras.io/getting-started/faq/#how-can-i-freeze-keras-layers)
  - Add output layer(s) for the PTB Diagnostic ECG Database and train the output layers.
  - Unfreeze the whole model.
  - Train the whole model.