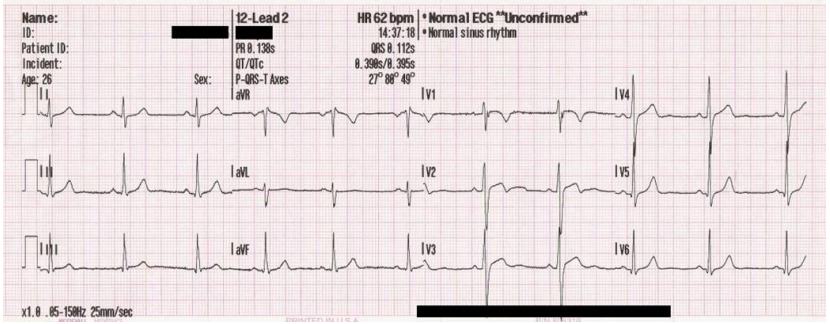
ML4HC Project 1: ECG time series

Laura Manduchi

Inspired by Cristobal Esteban slides.

ECG

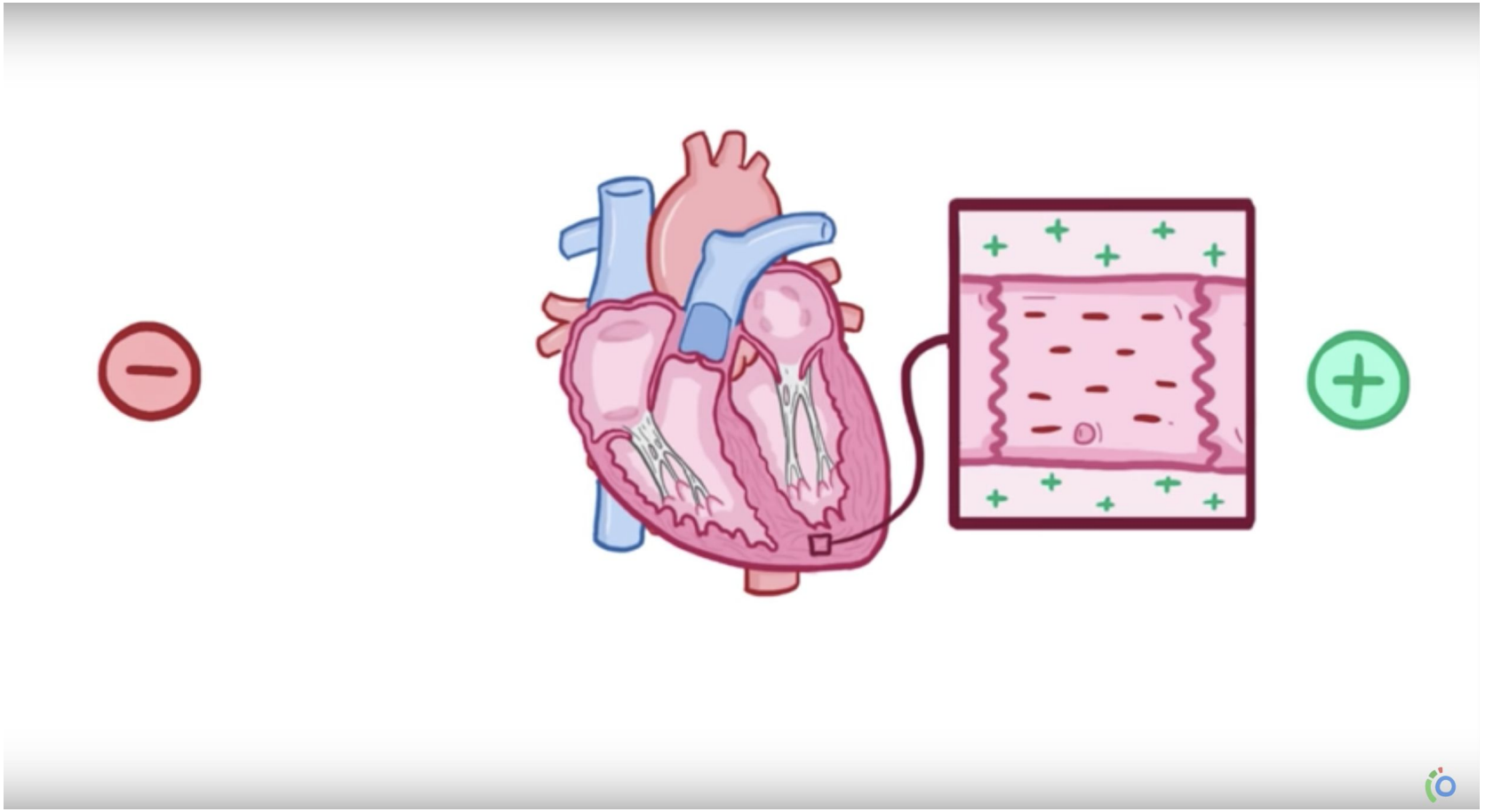
- **Electrocardiogram** (ECG or EKG) is a record of the electrical activity of the heart.
- **Electrodes**, placed on the skin of the patient, detect the small electrical changes of the heartbeat.



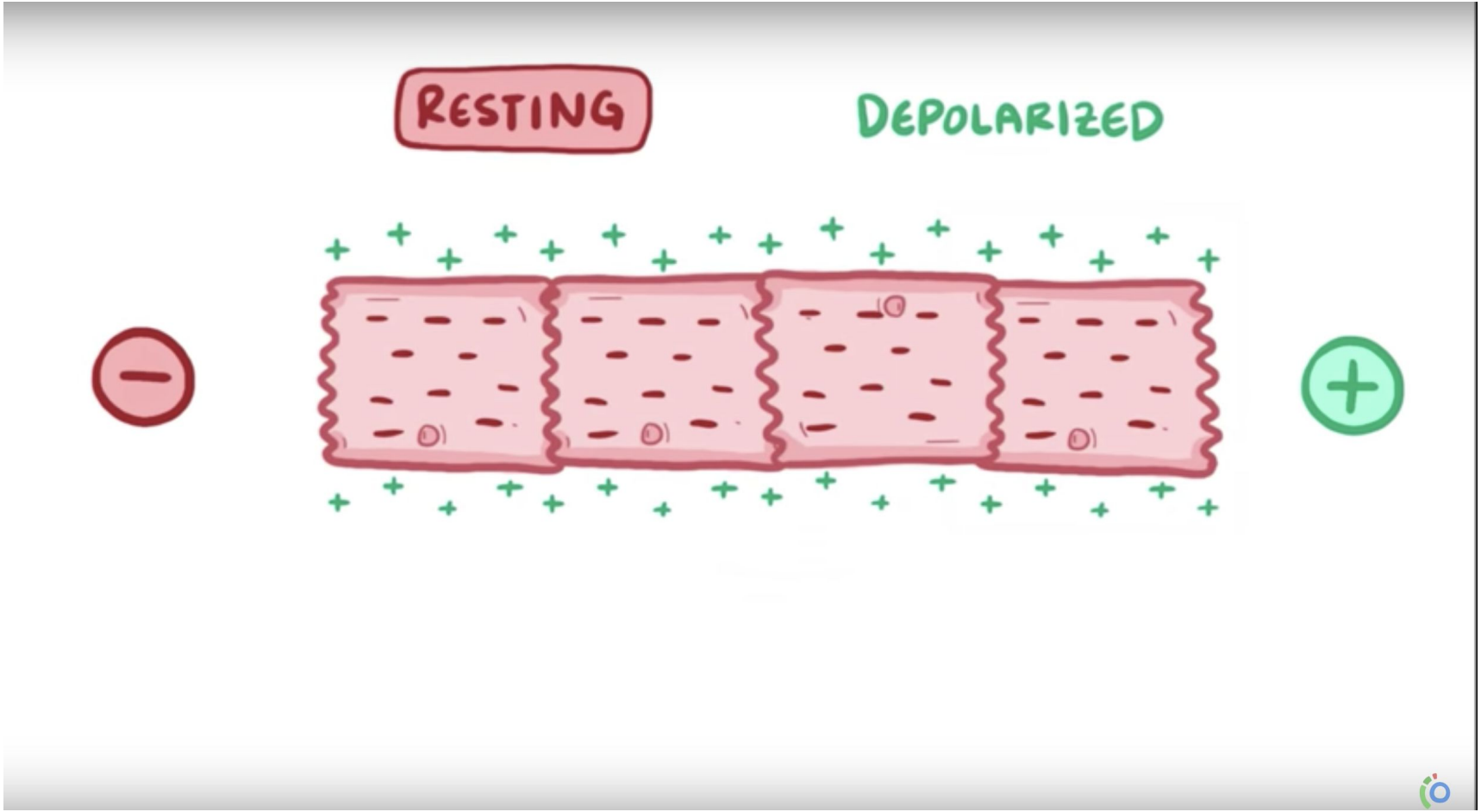
- **Electrical changes** are a consequence of cardiac muscle depolarization followed by repolarization during each cardiac cycle (heartbeat)
- **Depolarization** (biology) is a change within a cell, during which the cell undergoes a shift in electric charge distribution, resulting in less negative charge inside the cell.

ECG channels / leads

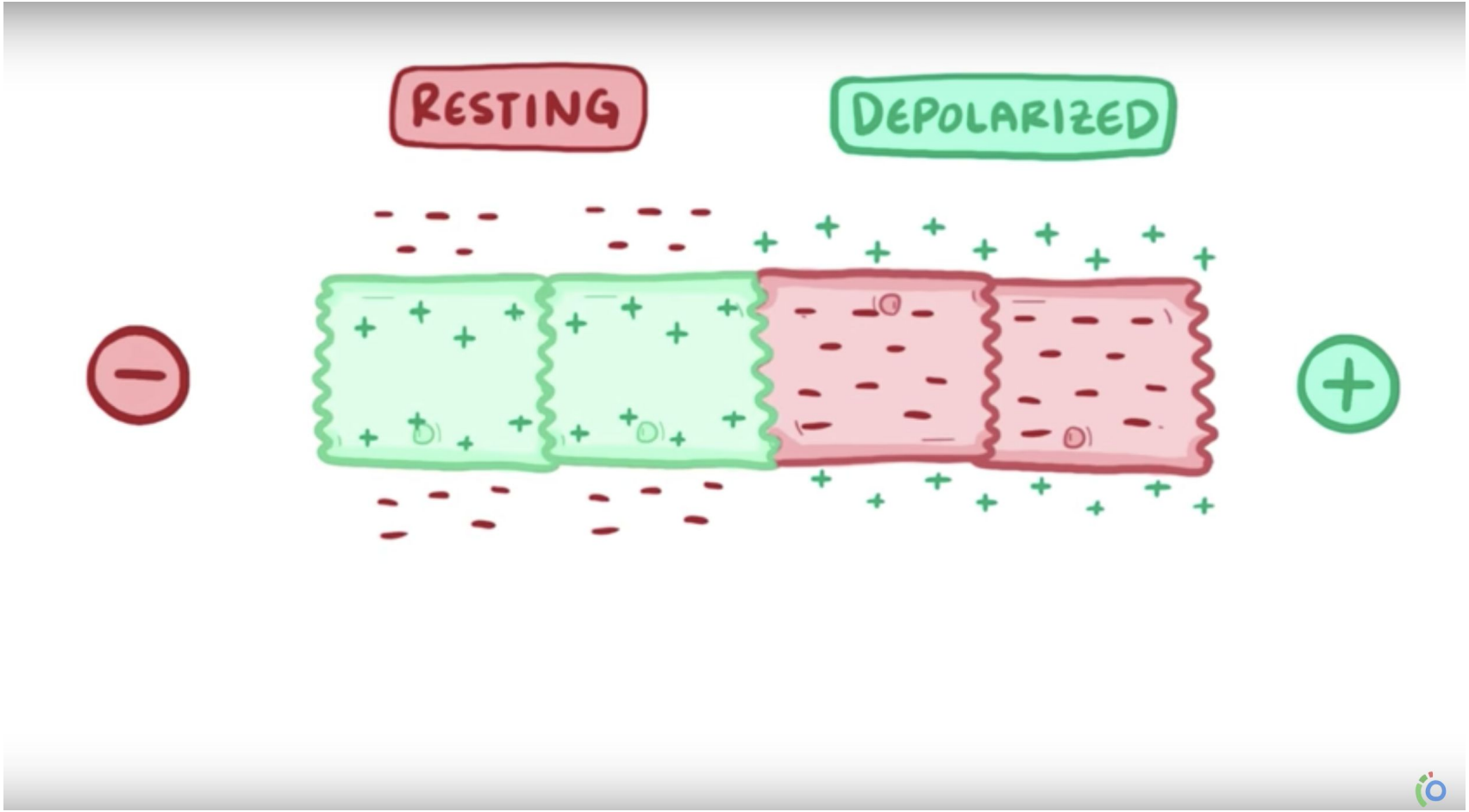
- A **lead** consists of two surface electrodes of opposite polarity or one positive surface electrode and a reference point.
- In a conventional 12-lead ECG, **10 electrodes** are placed on the patient's limbs and on the surface of the chest.
    - Electrodes are placed in standardized locations.
    - 12 different leads (angles) are used to record the overall magnitude of the heart's electrical potential over a period of time (usually ten seconds).



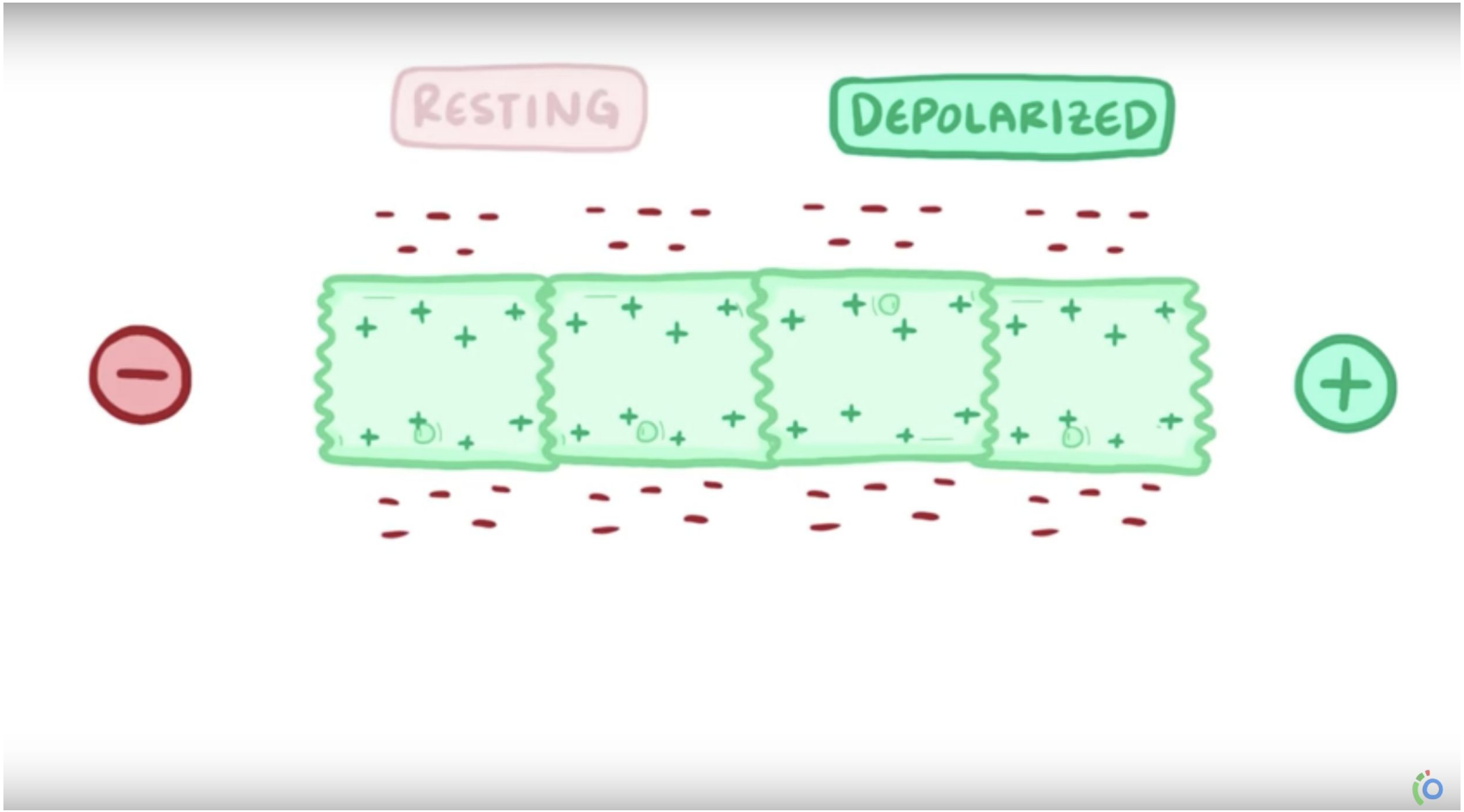
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



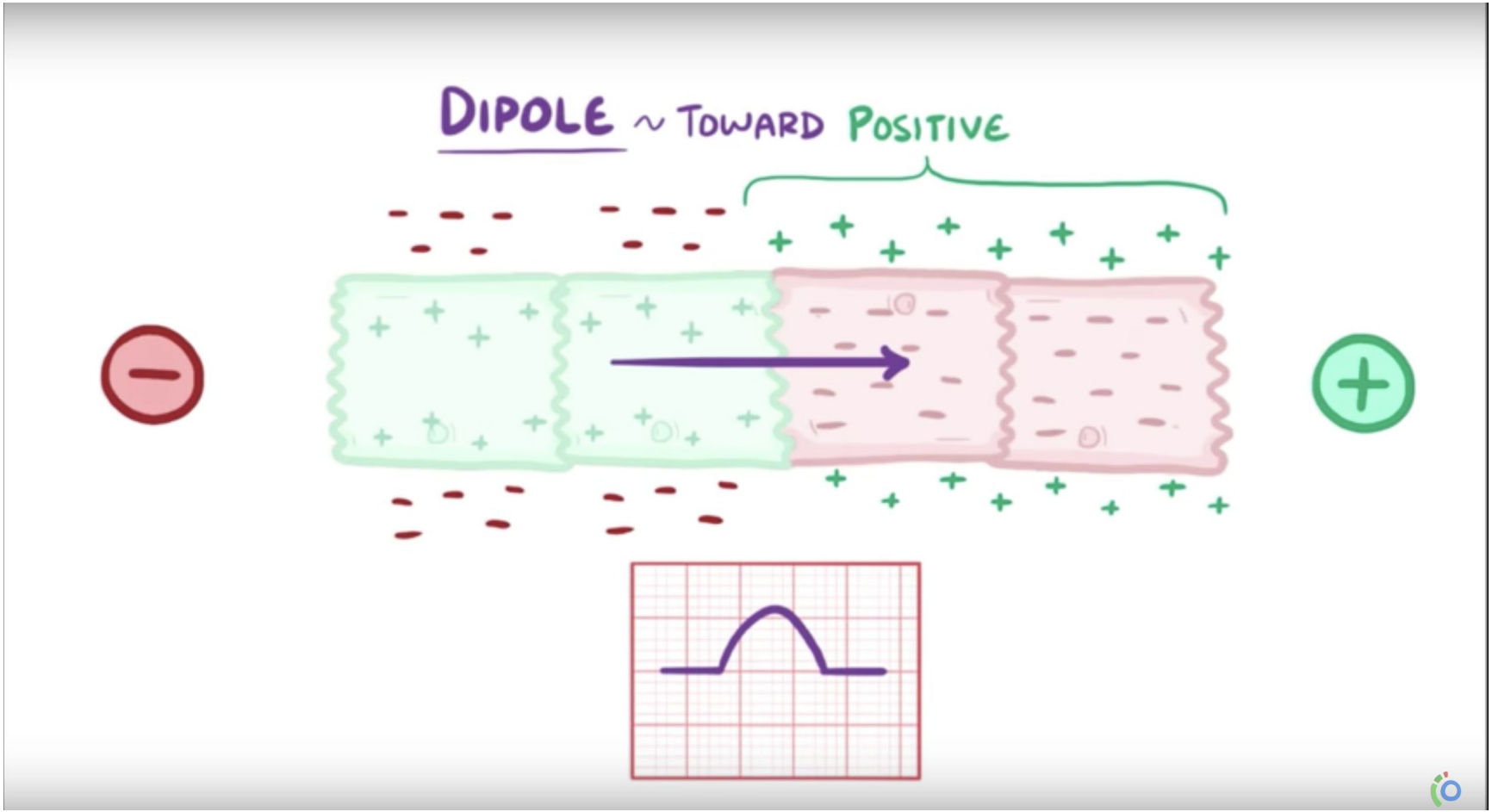
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



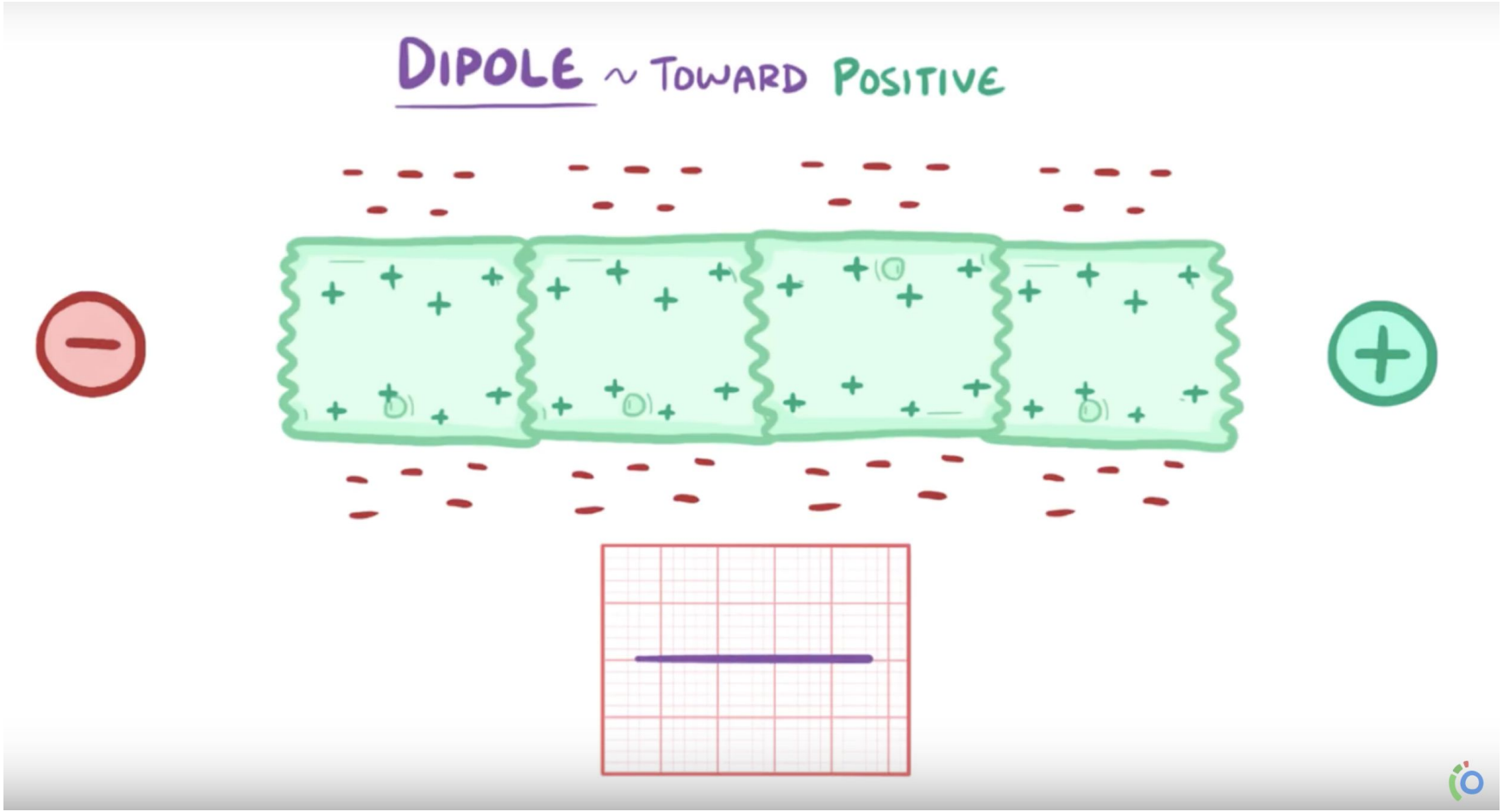
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



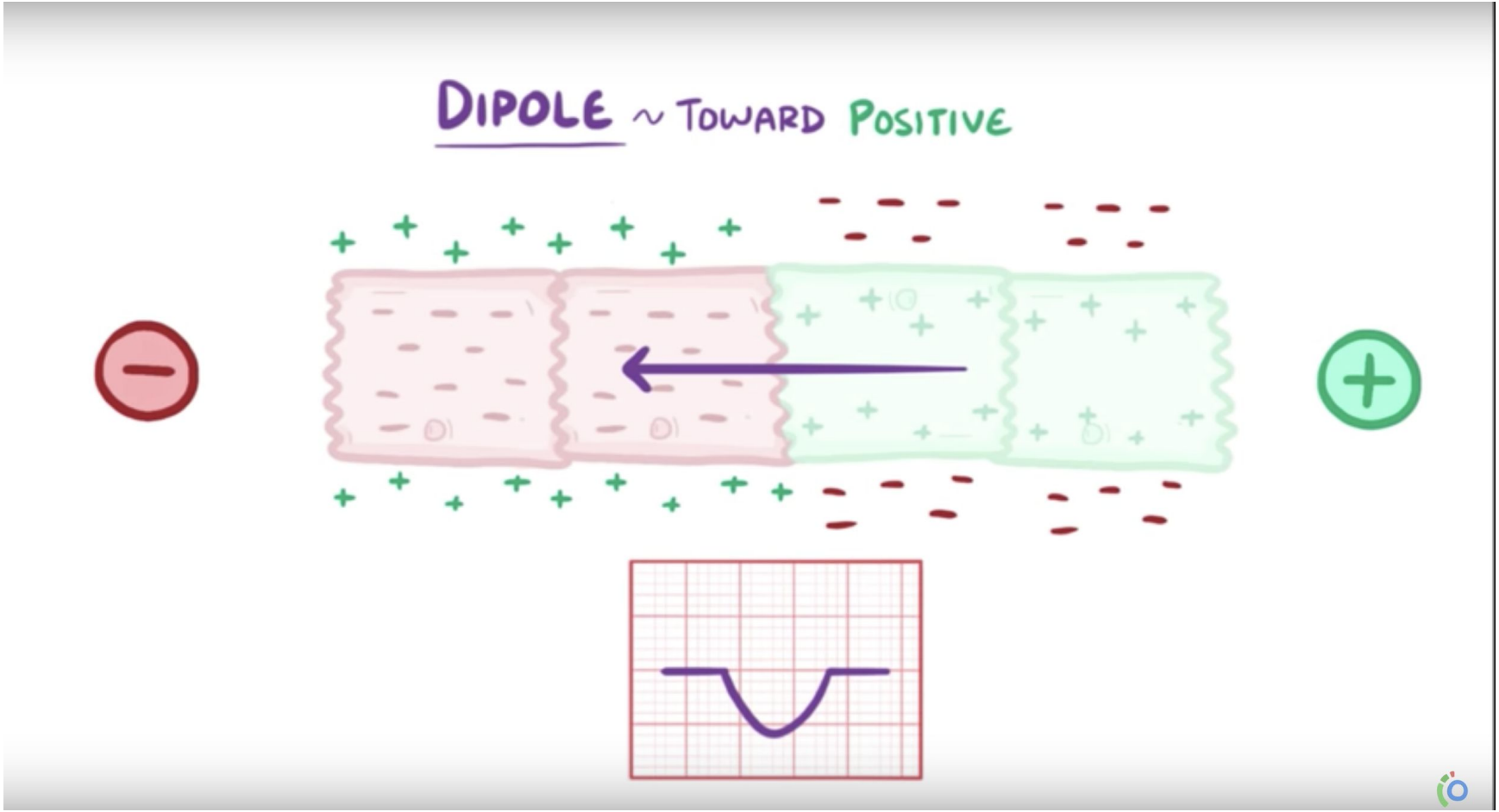
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



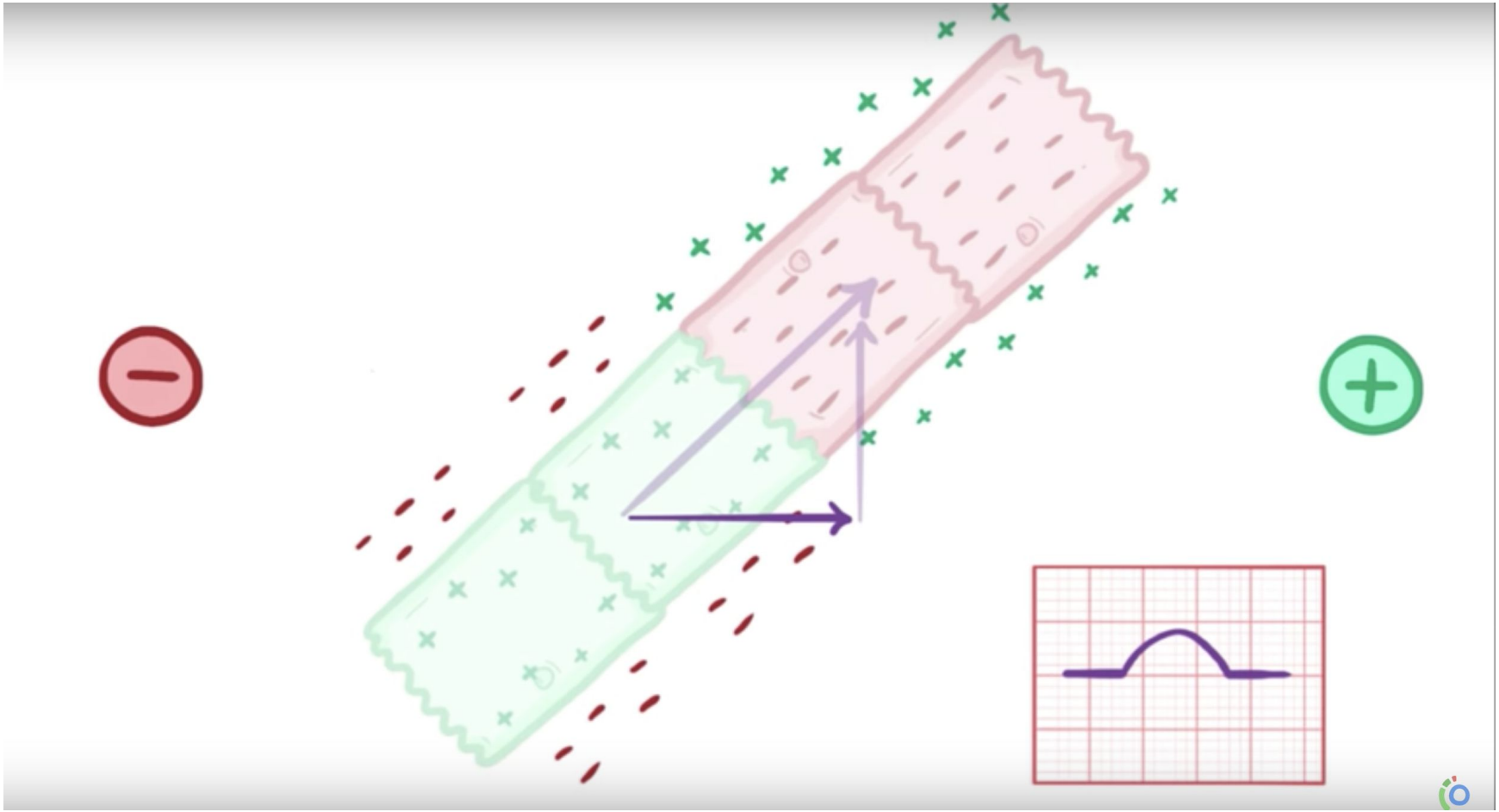
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



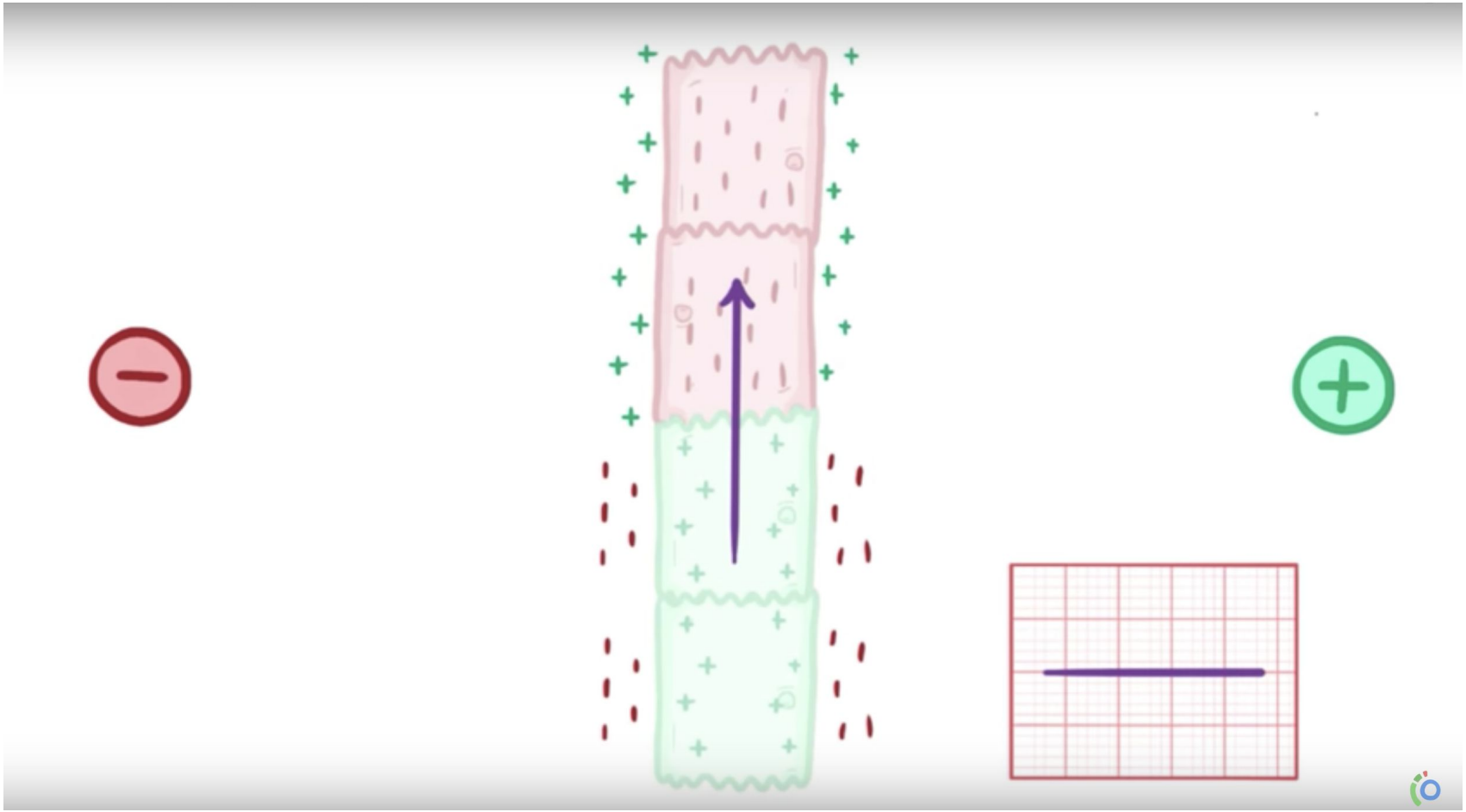
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



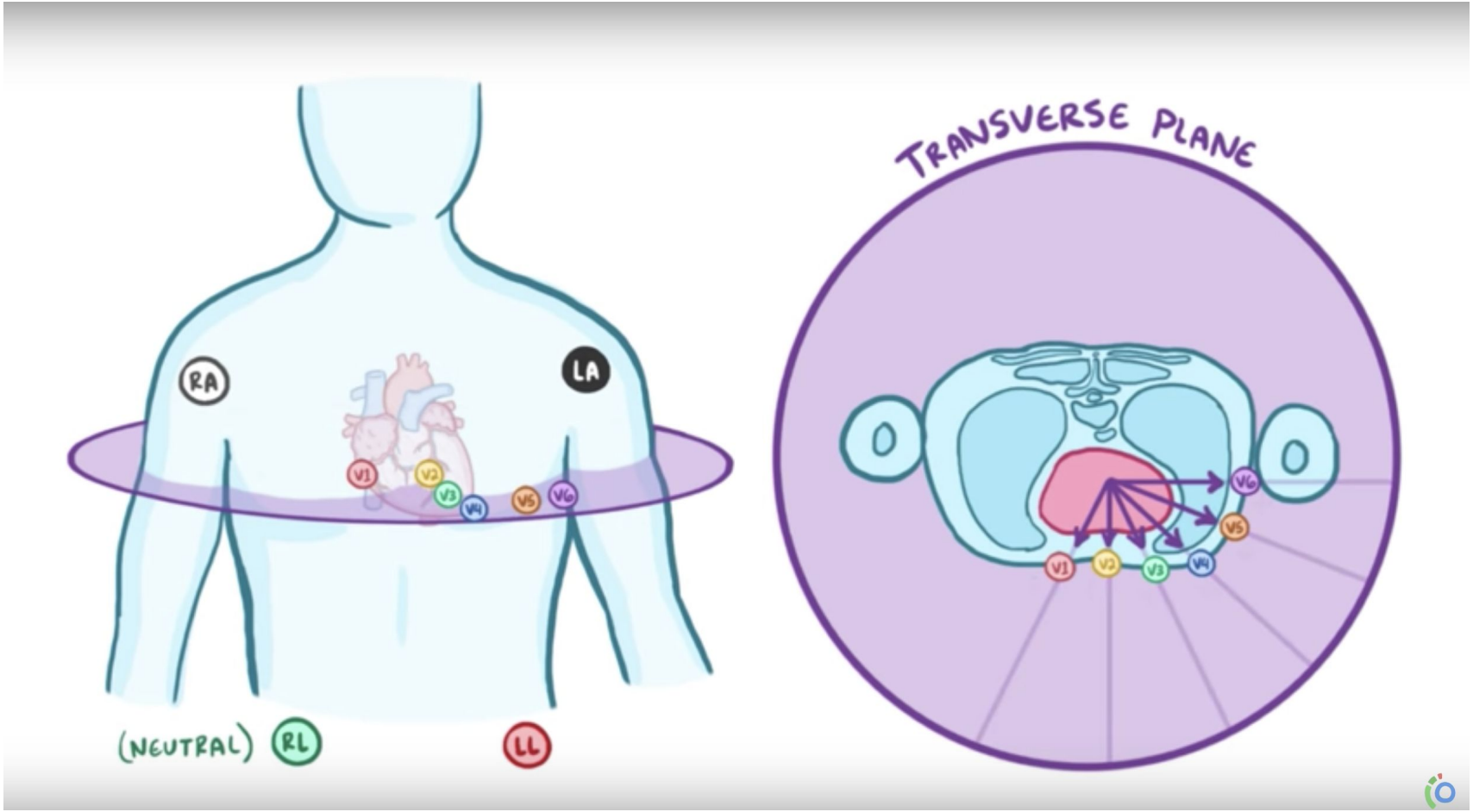
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



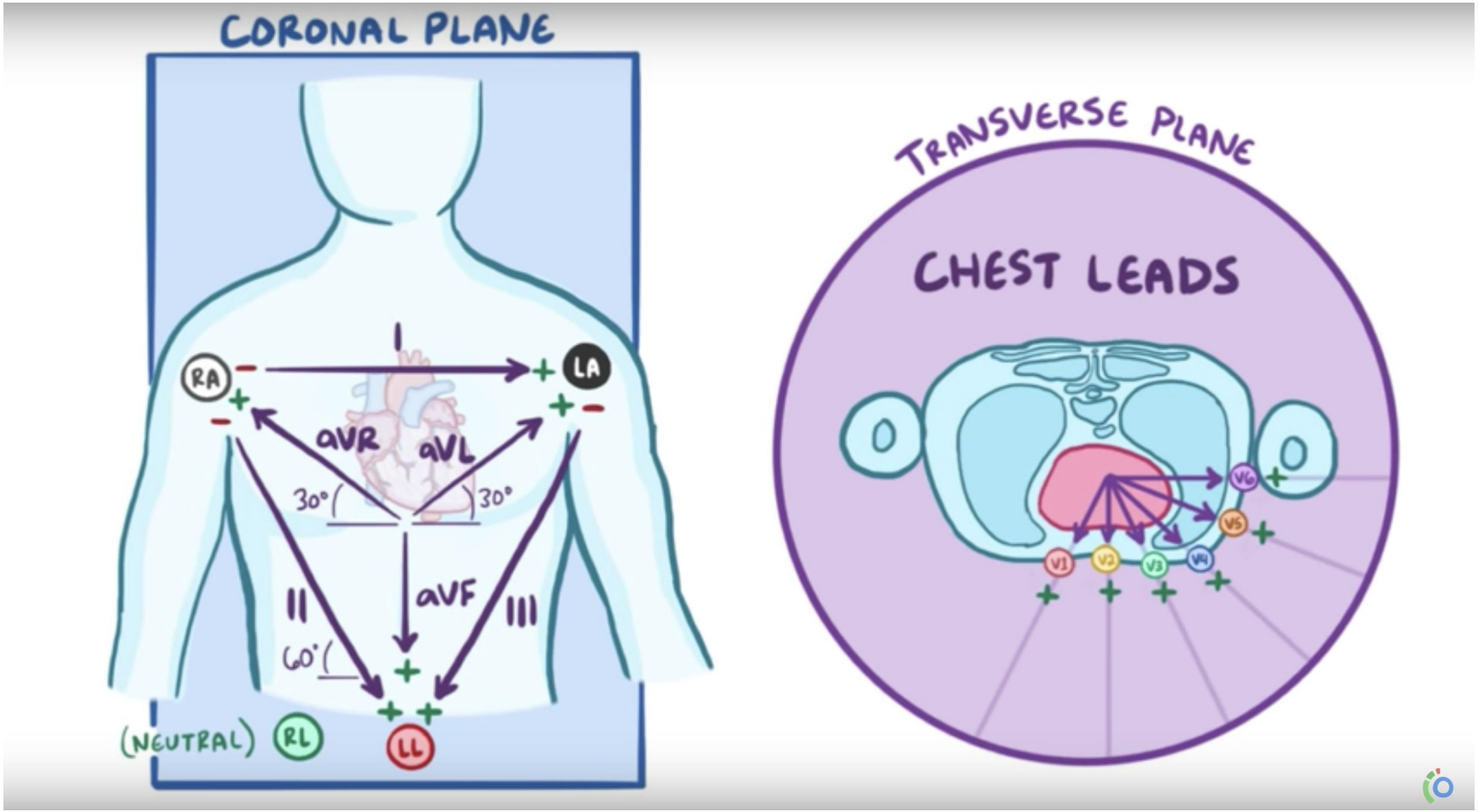
* from https://www.youtube.com/watch?v=xIZQRjkwV9Q



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



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



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

Motivation

- ECGs are very widely used as an inexpensive and noninvasive means of observing the physiology of the heart.
- Changes in the normal ECG pattern occur in numerous **cardiac abnormalities**, including cardiac rhythm disturbances, inadequate coronary artery blood flow, etc.
- **Personal devices** that do ECG will become more and more frequent -> more data available!
- **Machine Learning** systems will be needed to automatically evaluate all this data.

Apple watch ECG

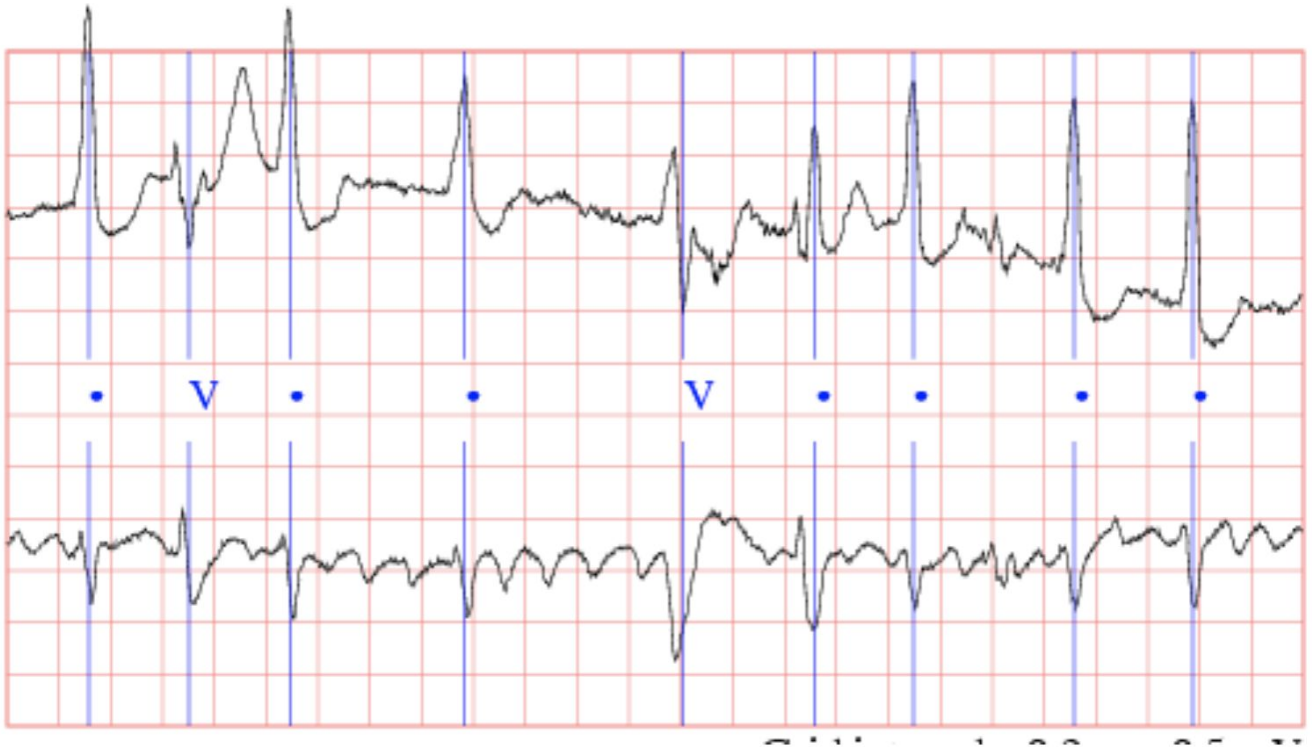
- **1 channel** ECG (it uses the electrical heart sensor to record the heartbeat).
- **Atrial Fibrillation** (AFib) occurs when the heart beats in an irregular pattern (the upper chambers of the heart beat out of sync with the lower chambers).
- Approximately 2% of people younger than 65 years old and 9% of people 65 and older have AFib.
- If left untreated, AFib can lead to **heart failure** or blood clots that may lead to **stroke**.
- **ECG app** accurately classifies an ECG recording into AFib and sinus rhythm.
- In a clinical trial of approximately 600 subjects, 99.6% specificity with respect to sinus rhythm classification and 98.3% sensitivity for AFib classification for the classifiable results.

MIT-BIH Arrhythmia Database

- https://physionet.org/physiobank/database/mitdb/
- First generally available set of standard test material for evaluation of arrhythmia detectors.
- **47 subjects**.
- 48 half-hour excerpts of **two-channel** ambulatory ECG recordings, 23 chosen at random and the rest were selected to include examples of uncommon but clinically important arrhythmias.
- 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) included with the database.

G. B. Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," in *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45-50, May-June 2001.

MIT-BIH Arrhythmia Database



The PTB Diagnostic ECG Database

- https://physionet.org/physiobank/database/ptbdb/
- 549 records from **290 subjects** (aged 17 to 87, mean 57.2).
- 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.
- 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.

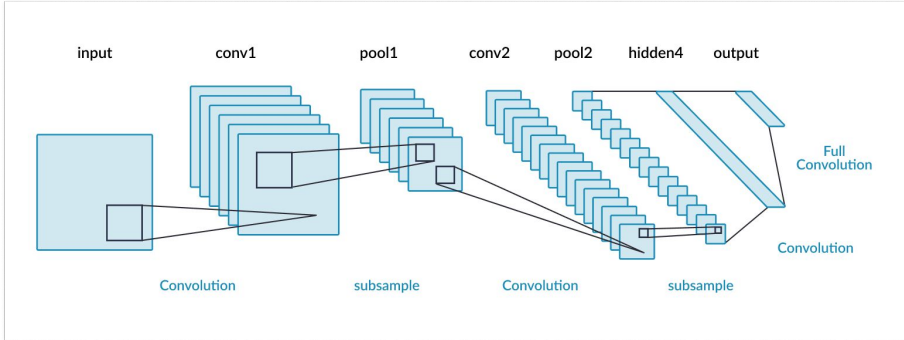
Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003). Circulation. 101(23):e215-e220.

The PTB Diagnostic ECG Database

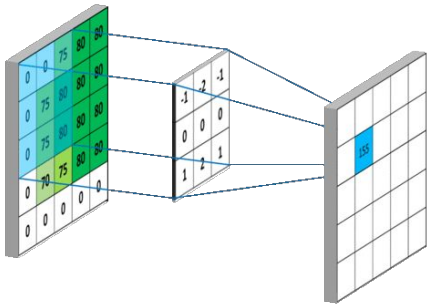
Diagnostic class	Number of subjects
Myocardial infarction	148
Cardiomyopathy/Heart failure	18
Bundle branch block	15
Dysrhythmia	14
Myocardial hypertrophy	7
Valvular heart disease	6
Myocarditis	4
Miscellaneous	4
Healthy controls	52

The PTB Diagnostic ECG Database

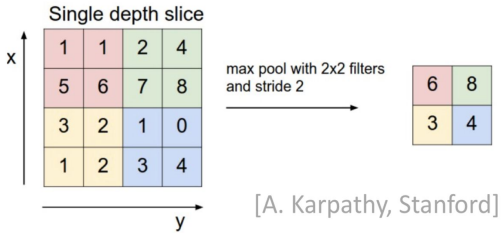
Background on CNN



Convolution



Subsampling



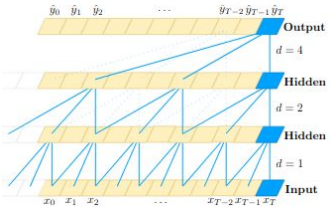
[A. Karpathy, Stanford]

- It captures the **spatial dependencies** of the data
- Reduces the number of parameters involved (**reusability of weights**)
- Encourages robustness against (small amounts of) **translation**

U. Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, Muhammad Adam, Arkadiusz Gertych, Ru San Tan, A deep convolutional neural network model to classify heartbeats, Computers in Biology and Medicine, Volume 89, 2017, Pages 389-396, ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2017.08.022.

1D-CNN vs RNN: advantages

- **Parallelism**. Unlike in RNNs where the predictions for later timesteps must wait for their predecessors to complete, convolutions can be done in parallel since the same filter is used in each layer.
- **Flexible receptive field size**. A CNN can change its receptive field size in multiple ways. CNNs thus afford better control of the model's memory size,and are easy to adapt to different domains.
- **Stable gradients**. Unlike recurrent architectures, CNN has a backpropagation path different from the temporal direction of the sequence. CNN thus avoids the problem.
- How to obtain a **Causal Architecture** ? Temporal Convolutional Network (TCN)



Bai, Shaojie & Kolter, J. & Koltun, Vladlen. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.

1D-CNN vs RNN: disadvantages

- **Data storage during evaluation**. In evaluation/testing, RNNs only need to maintain a hidden state and take in a current input xt in order to generate a prediction. In other words, the hidden state is a “summary” of the entire history and the actual observed sequence can be discarded. In contrast, TCNs need to take in the raw sequence up to the effective history length, thus possibly requiring more memory during evaluation.
- **Potential parameter change for a transfer of domain**. Different domains can have different requirements on the amount of history the model needs in order to predict.Therefore, when transferring a model from a domain where only little memory is needed (i.e., small k and d) to a domain where much longer memory is required (i.e.,much larger k and d), TCN may perform poorly for not having a sufficiently large receptive field.

Bai, Shaojie & Kolter, J. & Koltun, Vladlen. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.

Transfer Learning

- Deep learning approaches have a huge amount of variables which need to be trained on massive amounts of data.
- Medical data are often scarce.
- Solution: **knowledge transfer** between different tasks (datasets).
- Transfer learning and domain adaptation refer to the situation where what has been learned in one setting is exploited to **improve generalization** in another setting (Page 526, Deep Learning, 2016)
- A model trained for one task is re-purpose on a second related task.
- Two main approaches:
    - Weight initialization
    - Feature Extraction

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 based on CNNs.
- Two datasets:
  - **Arrhythmia Dataset**
  - **The PTB Diagnostic ECG Database**
- The model is first trained on the arrhythmia detection task (more labeled samples) and then the signal representation learned is successfully transferable to the MI prediction 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 Myocardial Infarction, 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.
- Samples are downsampled, normalized and padded.

ECG Heartbeat Classification: A Deep Transferable Representation

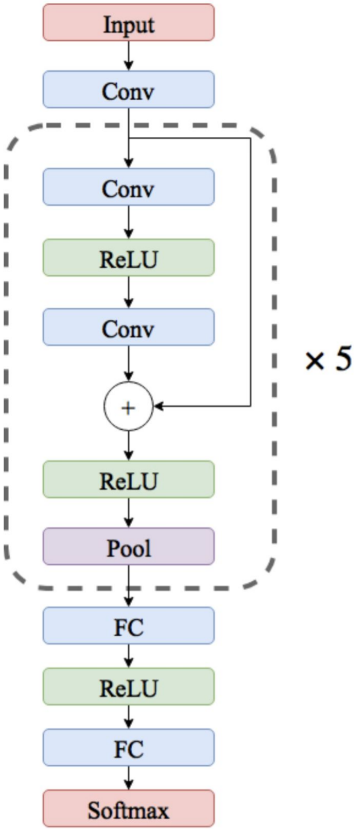


Fig. 2: Architecture of the proposed network.

- **Step 1:** Train the network on arrhythmia detection task
- **Step 2:** Take the model without the final Fully Connected layers and **freeze** it.
- **Step 3:** Add two fully connected layers for the binary prediction task.
- **Step 4:** Train the model on the myocardial infarction classification tasks, leaving the Conv part freezed.

NOTE: The entire model is used in Step 4 but only the last two layers are trained.

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

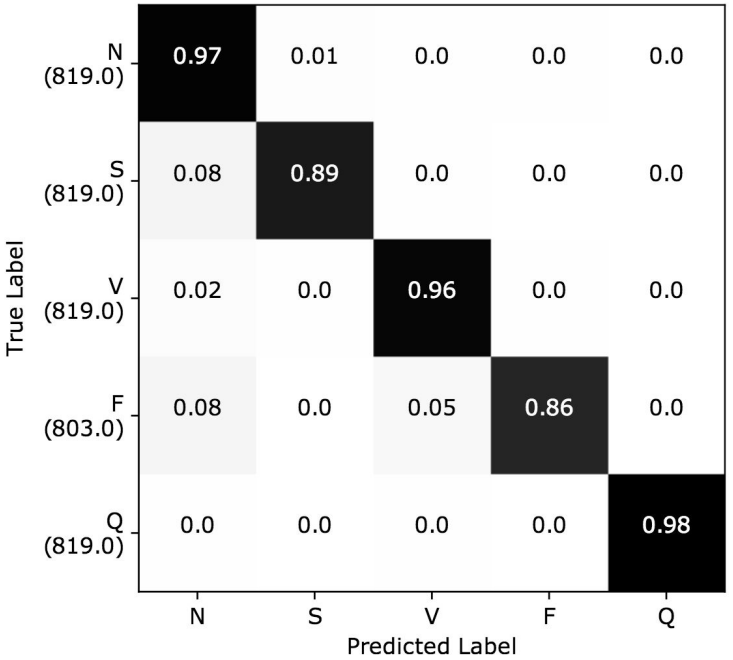


Fig. 3: Confusion matrix for heartbeat classification on the test set. Total number of samples in each class is indicated inside parenthesis. Numbers inside blocks are number of samples classified in each category normalized by the total number of samples and rounded to two digits.

Project Description:

- **Datasets (taken from kaggle)**:
  - **Arrhythmia Dataset:**
    - two different files (mitbih_train and mitbih_test)
    - 109446 samples and 5 classes ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]
    - [N: Normal beat, S: Supraventricular premature beat, V: Premature ventricular contraction, F: Fusion of ventricular and normal beat, Q: Unclassifiable beat]
  - **The PTB Diagnostic ECG Database:**
    - two different files (ptbdb_abnormal and ptbdb_normal)
    - 14552 samples and 2 classes (MI or normal)
  - All the samples are cropped, downsampled and padded with zeros if necessary to the fixed dimension of 188.
  - This dataset consists of a series of CSV files. Each of these CSV files contain a matrix, with each row representing an example in that portion of the dataset. The final element of each row denotes the class to which that example belongs.

Project Description:

- **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, these implementations doesn't contain residual blocks.
  - Don't take the transfer learning code of that repository.
- **Training / Testing split:**
  - **Important:** Use code of the baselines, see next slide

Train/Test split:

Arrhythmia Dataset

```
df_train = pd.read_csv("../input/mitbih_train.csv", header=None)
df_train = df_train.sample(frac=1)
df_test = pd.read_csv("../input/mitbih_test.csv", header=None)

Y = np.array(df_train[187].values).astype(np.int8)
X = np.array(df_train[list(range(187))].values)[..., np.newaxis]

Y_test = np.array(df_test[187].values).astype(np.int8)
X_test = np.array(df_test[list(range(187))].values)[..., np.newaxis]
```

https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_mitbih.py

The PTB Diagnostic ECG Database

```
df_1 = pd.read_csv("../input/ptbdb_normal.csv", header=None)
df_2 = pd.read_csv("../input/ptbdb_abnormal.csv", header=None)
df = pd.concat([df_1, df_2])

df_train, df_test = train_test_split(df, test_size=0.2, random_state=1337, stratify=df[187])


Y = np.array(df_train[187].values).astype(np.int8)
X = np.array(df_train[list(range(187))].values)[..., np.newaxis]

Y_test = np.array(df_test[187].values).astype(np.int8)
X_test = np.array(df_test[list(range(187))].values)[..., np.newaxis]
```

https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_ptbdb.py

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 **baselines**.
- 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: 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: 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.

Deliverables

- Report of max. 2 pages.
- 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.
- Use Keras functional API.
- Do not hardcode any results!
- Again: sequential execution and **reproducibility** !!
- **Deadline**: 18.03.2020

Hyperparameter search

Part of the training set should be used as validation set. (train/validation/test 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).

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
```

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)
```

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.

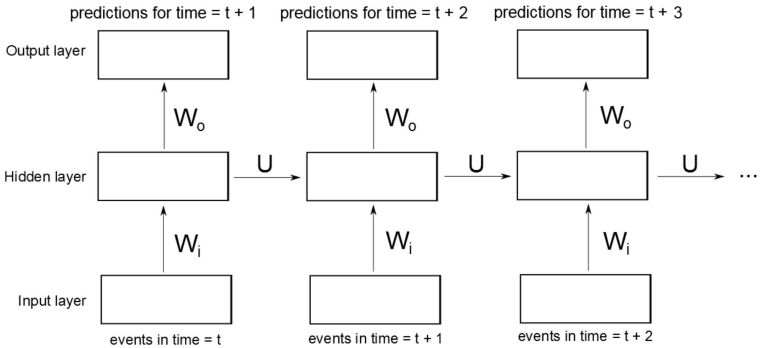
Example

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.

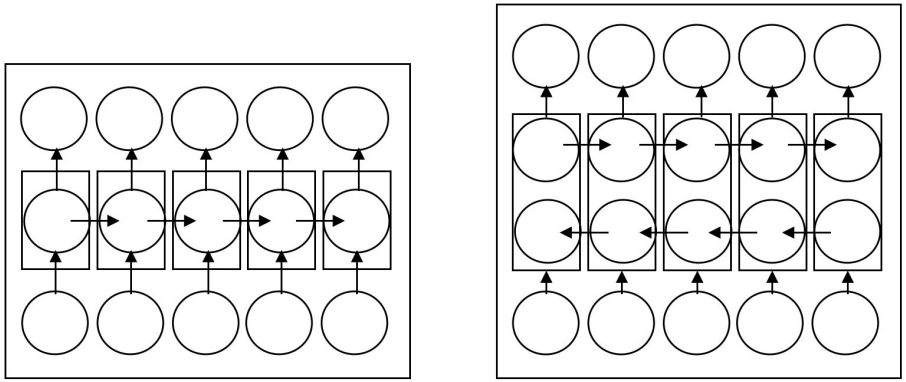
The input should be at least 3D, and the dimension of index one will be considered to be the temporal dimension.

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')
```