Conversion to MEX Vocabulary: Auto-generation of Code from Deep Learning Research Papers

NATURAL LANGUAGE PROCESSING LAB

SAMIN PAYRO

SUPERVISOR: DIEGO ESTEVES

MEX Vocabulary

Many machine learning experiments have been published → Large output data → Difficult to analyze and archive properly without provenance metadata

MEX:

- **lightweight and flexible schema** for publishing machine learning outputs metadata.
- Method to describe experiments
- Makes further analysis and integrations easier.
- Focus is on practical and important aspects (e.g. input parameters).
- Avoids more sophesticated parameters (e.g. optimization) → Keep it simple.

Parts of MEX

:**Tool**: the software tool used by the experiment

:ToolParameter (:ToolParameterCollection): parameters specific defined for the tool

:Algorithm: the algorithm

:HyperParameter (:HyperParameterCollection): the set of hyperparamters for a given algorithm

:AlgorithmClass: the class of an algorithm (e.g.: Decision Trees or Support Vector Machines)

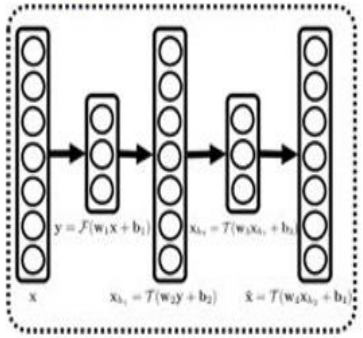
Why to do this auto-generation?

With an abundance of research papers in deep learning → reproducibility or adoption of the existing works becomes a challenge.

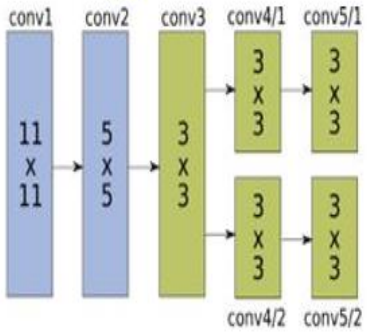
Many papers do not come with source code → It‘s hard to compare various papers or manipulate the codes.

Various types of deep learning flows

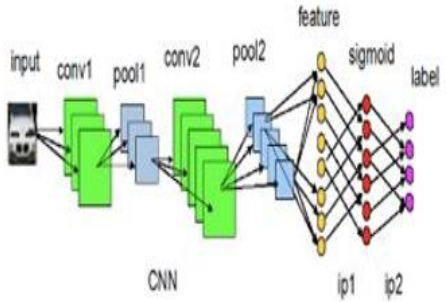
(a) Neurons Plot



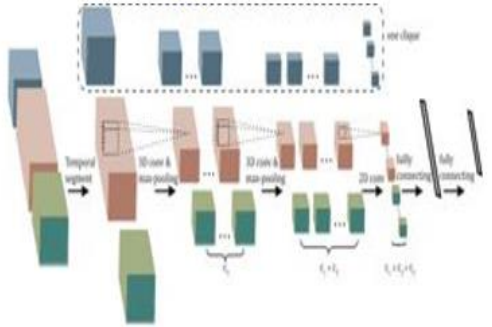
(b) 2D Box



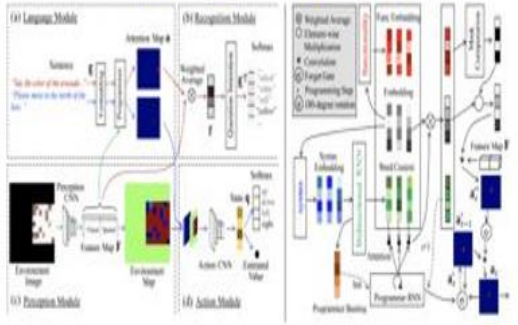
(c) Stacked2D Box



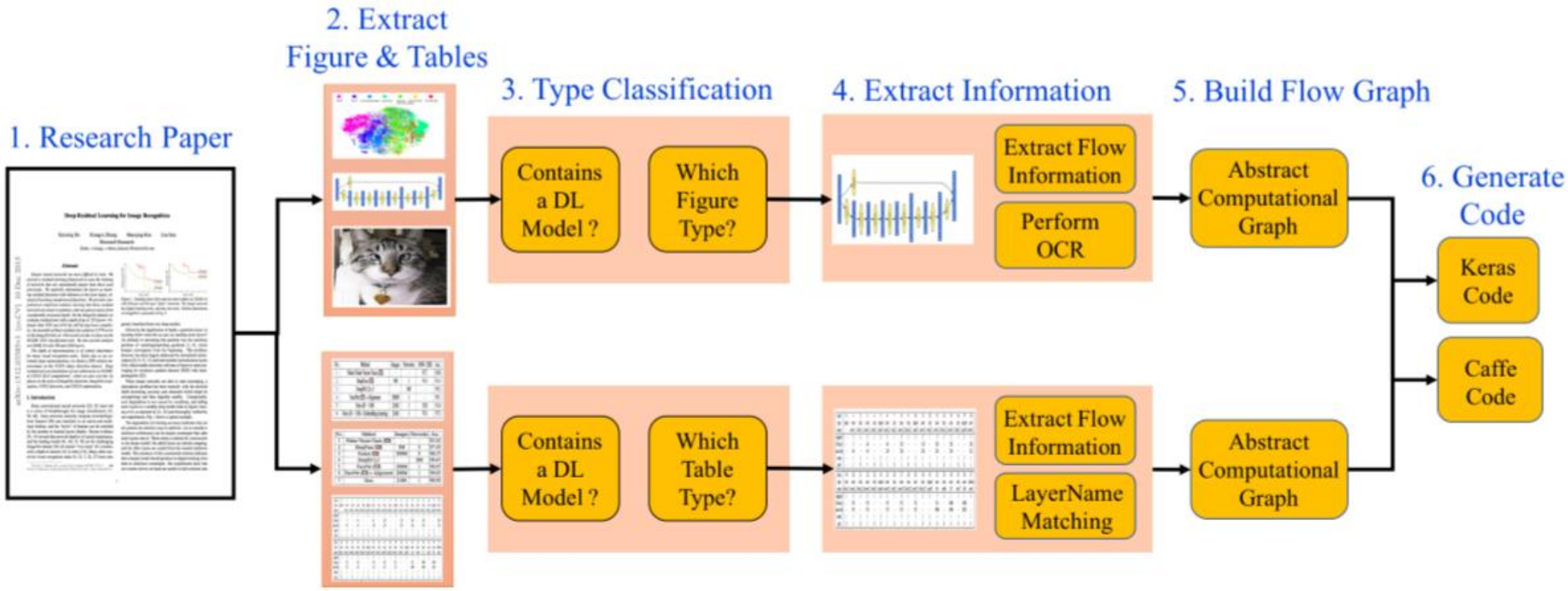
(d) 3D Box



(e) Pipeline plot



How to do the auto-generation?



Computational graph (JSON format)

Abstract Computational Graph

```
{
  "layer_type" : "Pooling2D",
  "layer_name" : "MaxPool1",
  "layer_params" : {
        "function" : "MAX",
        "trainable" : true,
        "stride_row" : 2,
        "kernel_col" : 2,
        "kernel_row" : 2,
        "stride_col" : 2,
        "border_mode" : "valid"
  }
  “from_layers”: [“Conv1”]
}
```

Caffe Protobuf

```
layer {
  name: "MaxPool1"
  type: "Pooling"
  bottom: "Conv1"
  top: "MaxPool1"
  pooling_param {
        pool: MAX
        kernel_size: 2
        stride: 2
  }
}
```

Keras Python

```
MaxPooling2D(
        pool_size=(2, 2),
        strides=(2,2),
        padding='valid',
        name='MaxPool1'
)(Conv1)
```