ALGORITHMIC DIFFERENTIATION

BITS F276 – Design Oriented Project
28/4/16

INTRODUCTION

Traditional Approaches

- Symbolic Differentiation
- Method of Finite Differences
- Complex Taylor Series Expansion

ALGORITHMIC DIFFERENTIATION

- AD works on the principle of Chain Rule at the operator level.

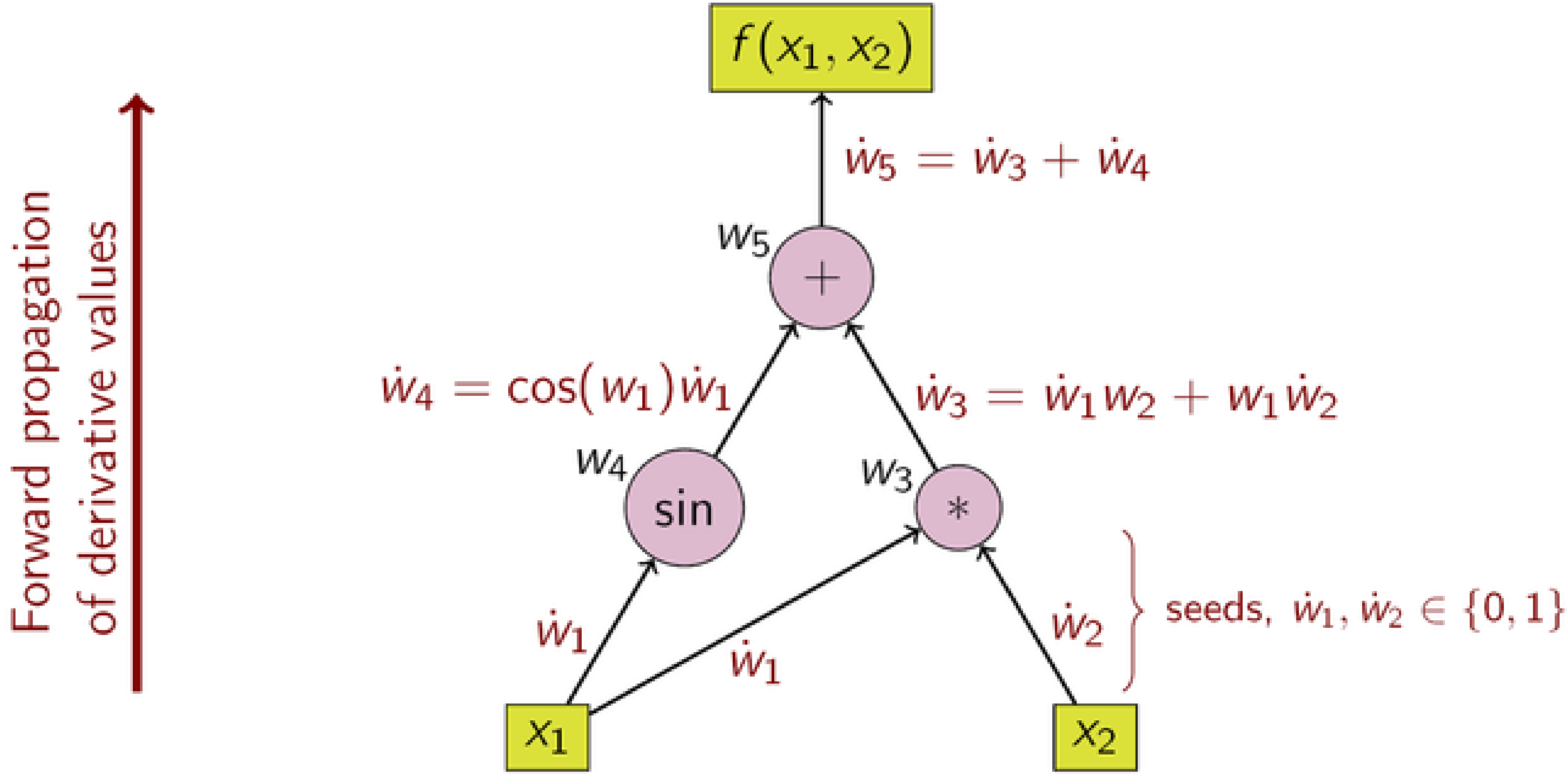
$$f = f_1\left(f_2\left(f_3 \ldots f_n(x)\right)\right)$$

We need,

- $$\frac{\partial f}{\partial x} = \frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_n} \cdot \frac{\partial f_n}{\partial x}$$

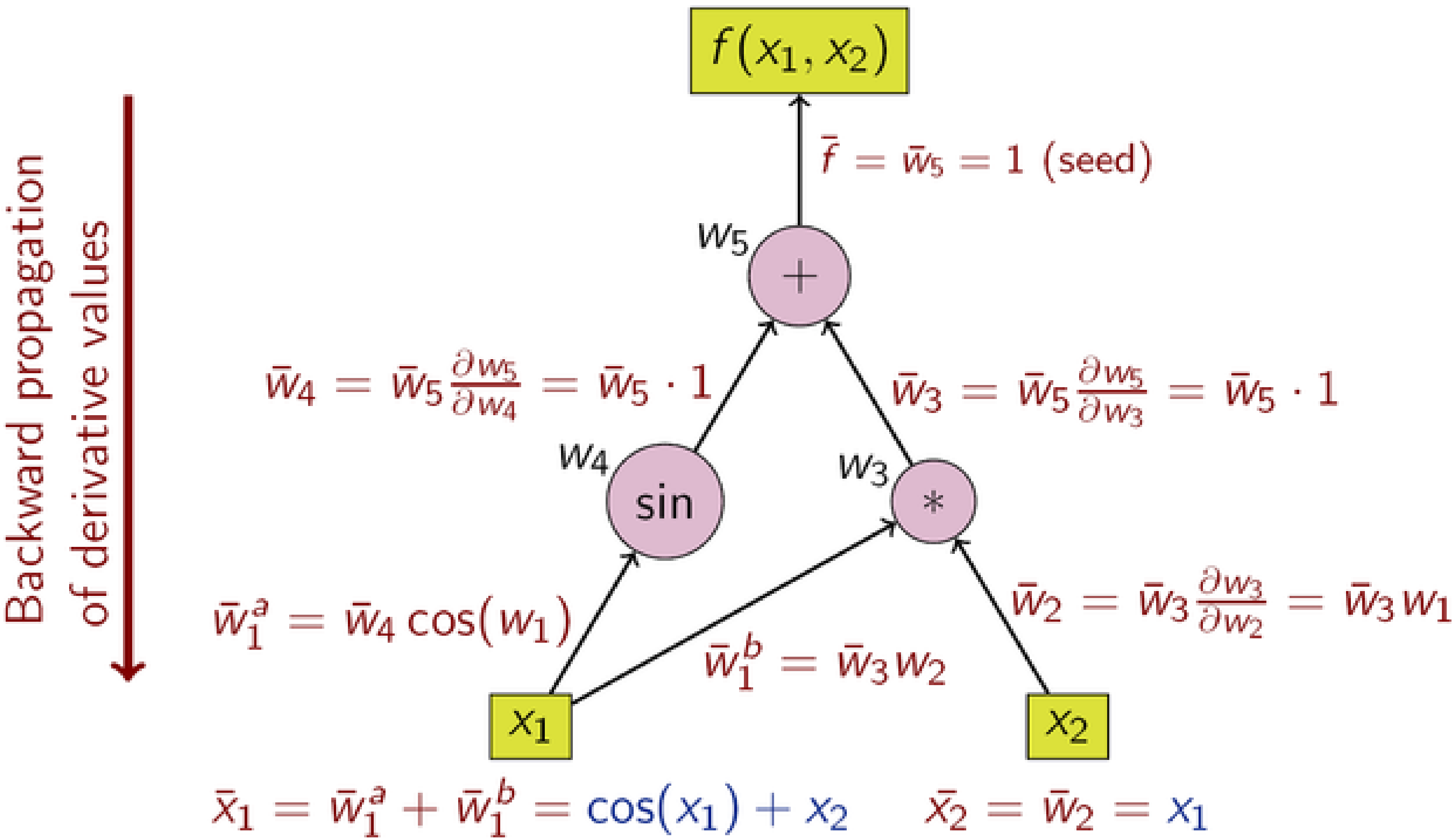
FORWARD MODE

- Forward Mode refers to the forward accumulation of the gradient information. i.e. right to left traversal of the chain rule.



REVERSE MODE

- Reverse mode refers to the reverse accumulation of the gradients. I.e right to left traversal of the chain rule.



COMPARISON OF MODES

Forward Mode requires **n** passes to evaluate the derivative whereas Reverse Mode requires **m** passes. Therefore, for m >> n, Reverse Mode is preferred and Forward Mode for n >> m.

Criteria to be considered:

- Complexity
- Memory

The problem of computing a full Jacobian of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with a minimum number of arithmetic operations is known as the *optimal Jacobian accumulation (OJA)* problem, which is NP-complete under mild assumptions.

COMPUTATIONAL GRAPHS

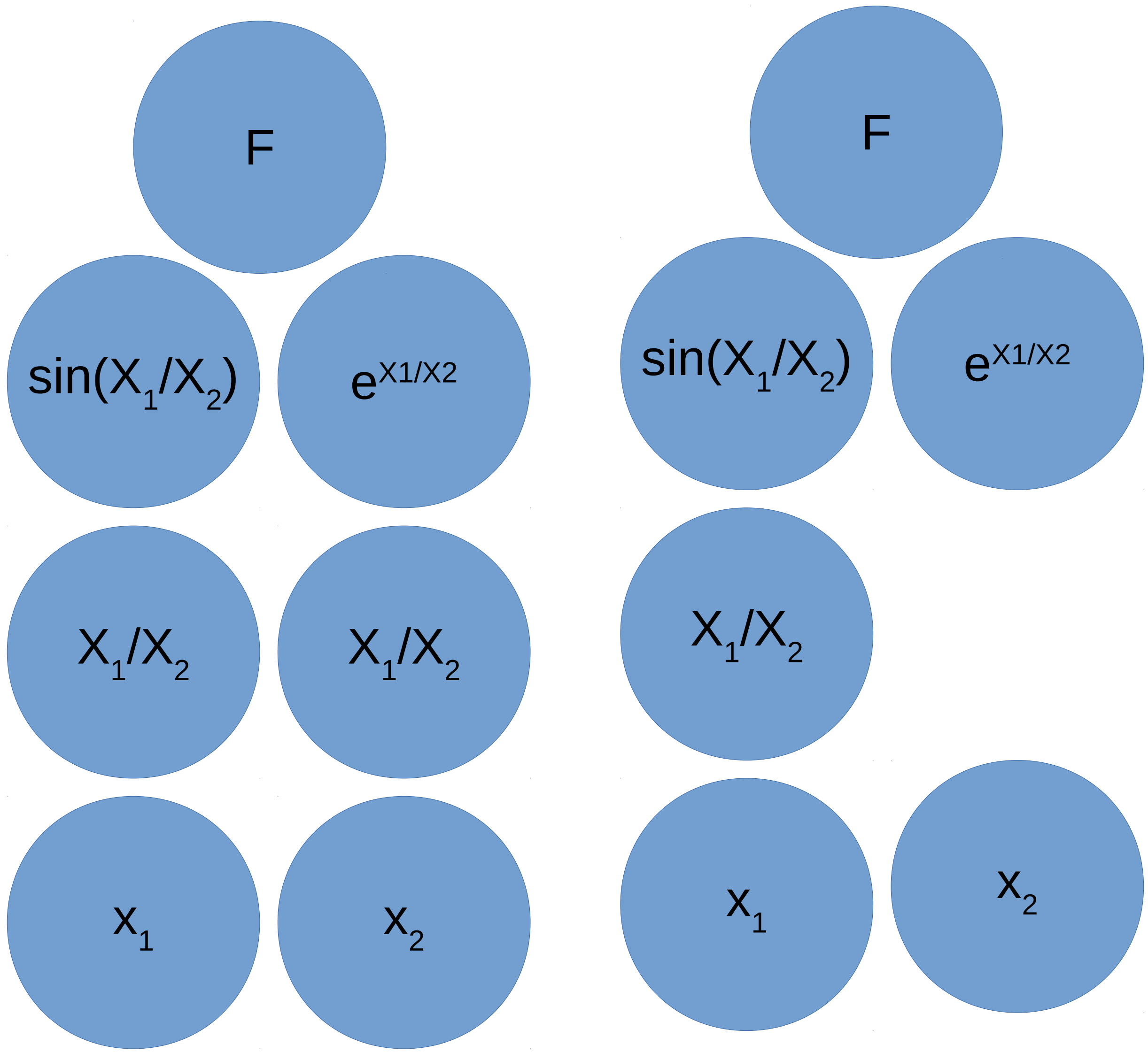
Definition:

$$G(F) = (V, E)$$

$$V = \{ V_x, V_y, V_z\}$$

- V_x : set of vertices representing independent variables.
- V_y: set of vertices representing intermediate variables.
- V_y: set of vertices representing output varibales.

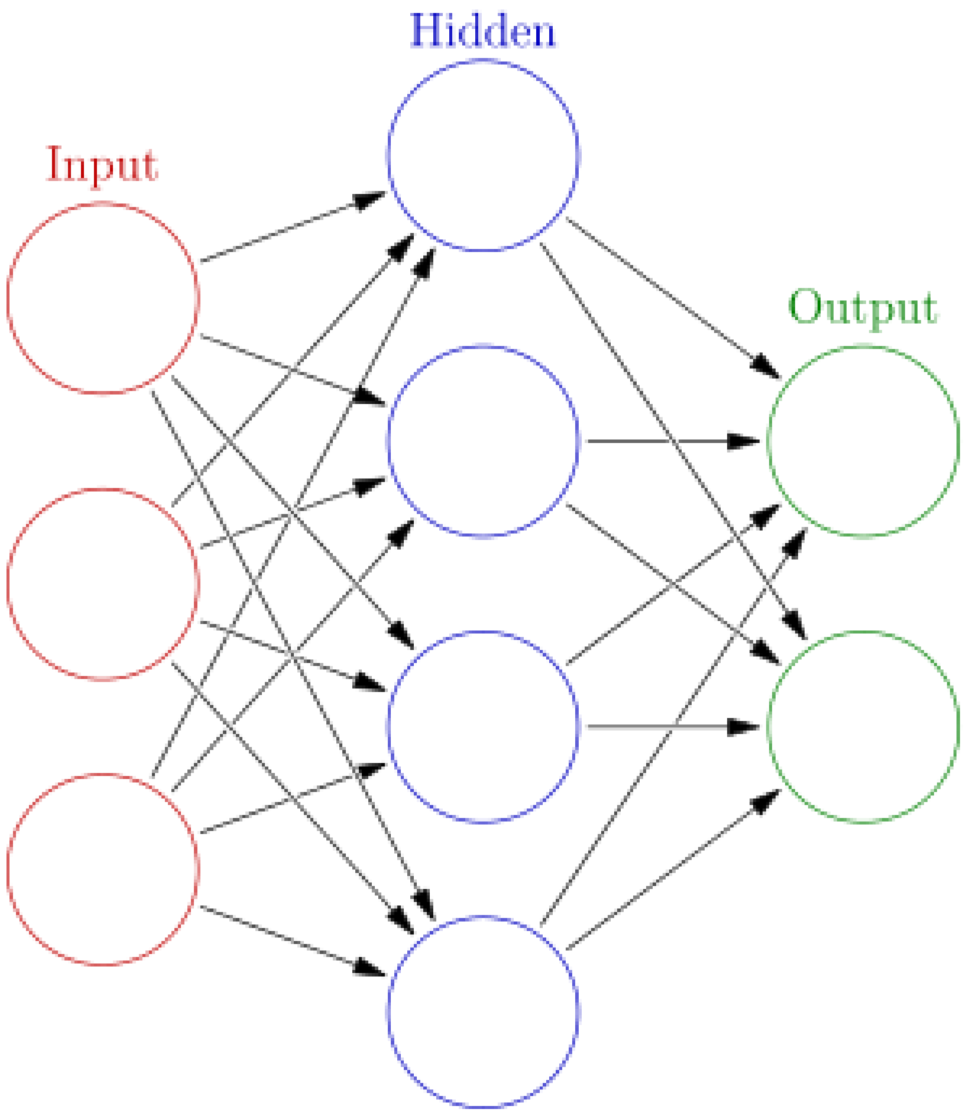
GRAPH OPTIMISATION



GRAPH OPTIMISATION

- Edge Separator
- Matrix Chaining
- Vertex Elimination

NEURAL NETWORKS



NEURAL NETWORKS

$$f_l(X) = \sum_D \sum_j \sigma\left(\sum_i w_{ij} x_i\right)$$

For each hidden layer l

$$E = \left(\sum_D \left(\sum_j \sigma\left(\sum_i w_{ij} x_i\right)\right) - y_d\right)^2$$

Therefore, to compute accurate synapse weights we need to find gradient of E w.r.t. to W.

$\frac{\partial E}{\partial w_{ij}}$ For all i, j in each layer.

CATEGORIES

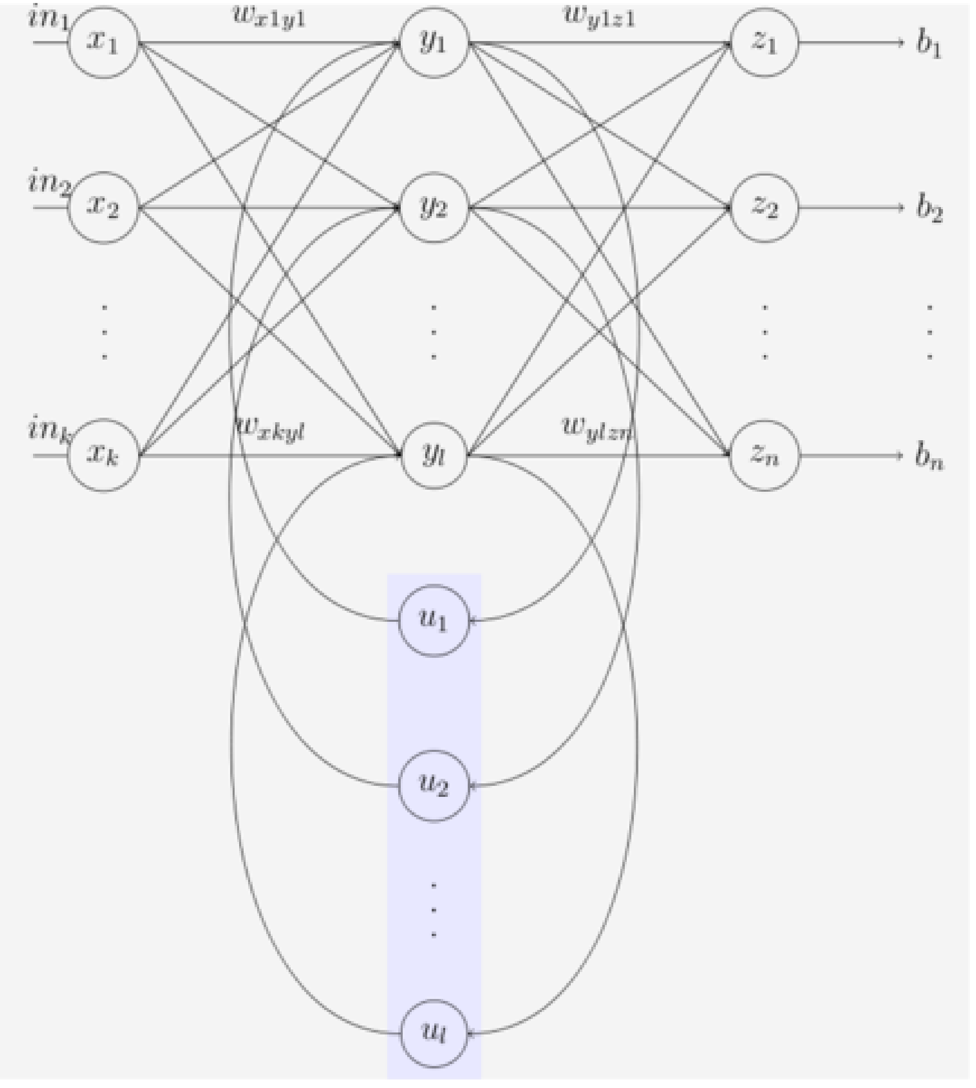
Feedforward NN

- Trained using backprop:

$$w_{ij} \leftarrow w_{ij} + \eta \frac{\partial E}{\partial w_{ij}}$$

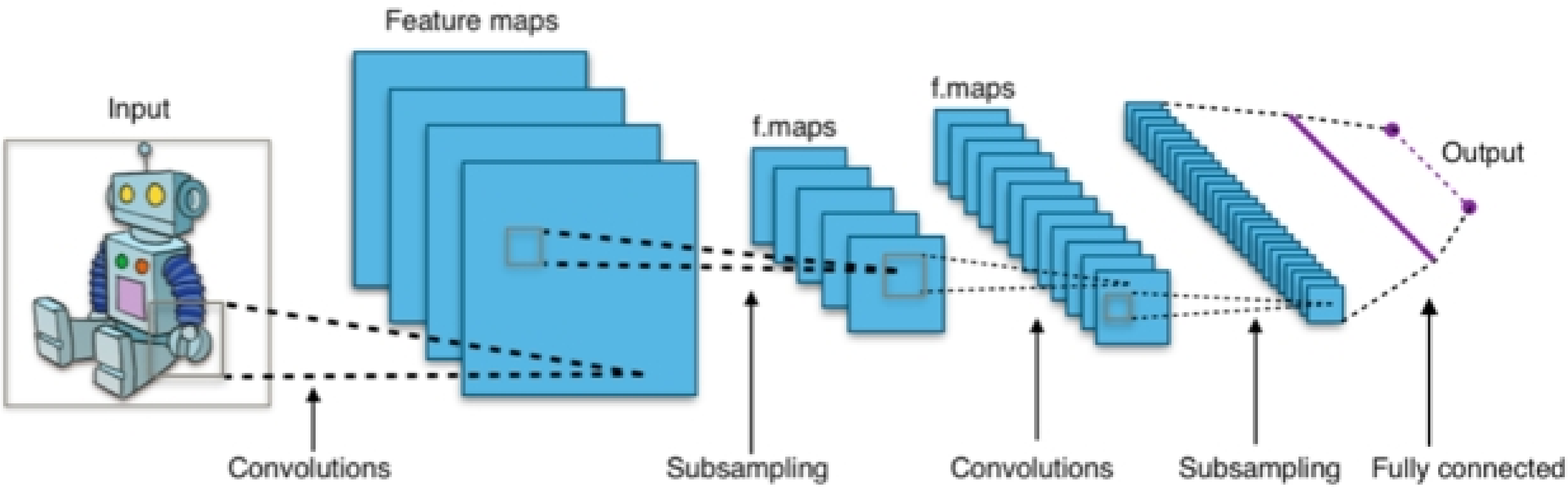
CATEGORIES

Recurrent NN



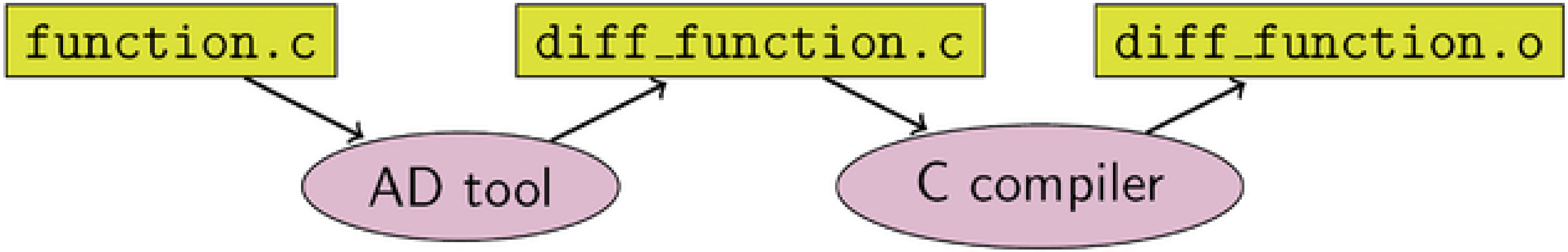
CATEGORIES

- Convolutional NN



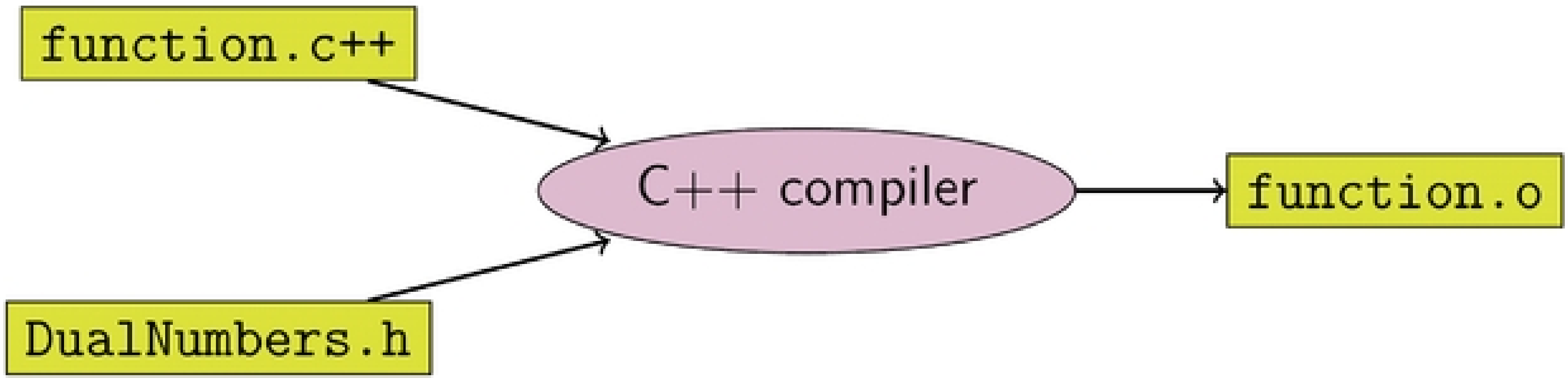
IMPLEMENTATION

- Source Code Transformation



IMPLEMENTATION

- Operator Overloading



IMPLEMENTATION

Rationale

- Operator Overloading easier to implement given binary nature of operator and parsing of function code.
- Intuitively easier to program as Derivative code is appended to normal evaluation code as in-built semantic rules for the parser.
- Allows convenient construction of syntax tree. Therefore easier to generate Computational Graph / Evaluation Trace.

IMPLEMENTATION

Codebase design paradigm

Variables – New data type defined for necessary attributes.

Memory – Checkpoint methods for Taping Schemes specific to Neural Networks.

IMPLEMENTATION

Optimisation

- Parser based – Performed at compile time alongside other code optimisation.
- Reference Count – Repeatedly used variables flagged to decrease memory usage.
- Evaluation Sweep – Clearing Trace at appropriate intervals to efficiently utilise memory.
- Graph Methods – Vertex Elimination / Matrix Chaining.

SCOPE AND LIMITATIONS

- Hessian based Gradient Descent – Second Derivative optimisation algorithms.
- Activation Functions – Treat the activation functions of each neural layer as an elementary function.
- GPU enhanced Computations – Where Function evaluation can be parallelised, so can AD tool. Devise efficient hybrid sweep schemes.

SCOPE AND LIMITATIONS

- High Dimensionality – requires distributed processing

- Performace Loss for Basic Network architectures. Hand-Written derivative code more efficient.

CONCLUSION

- The techniques discussed above provide guarantees on the reduction of computational complexity and memory efficiency over a general purpose AD tool.
- Reduce Prototyping time.
- Produce more legible, structured code.

REFERENCES

Recent Advances in Algorithmic Differentiation - Shaun Forth, Paul Hovland, Eric Phipps, Jean Utke, Andrea Walther

Evaluating Derivatives – Andreas Griewank

Automatic differentiation in machine learning: a survey – Baydin et al.

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

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

THANK YOU