CATEGORICAL FOUNDATIONS OF GRADIENT-BASED LEARNING

(CRUTTWELL, GAVRANOVIĆ, GHANI, WILSON, ZANASI)

GOAL:

PROVIDE A CATEGORICAL FRAMEWORK

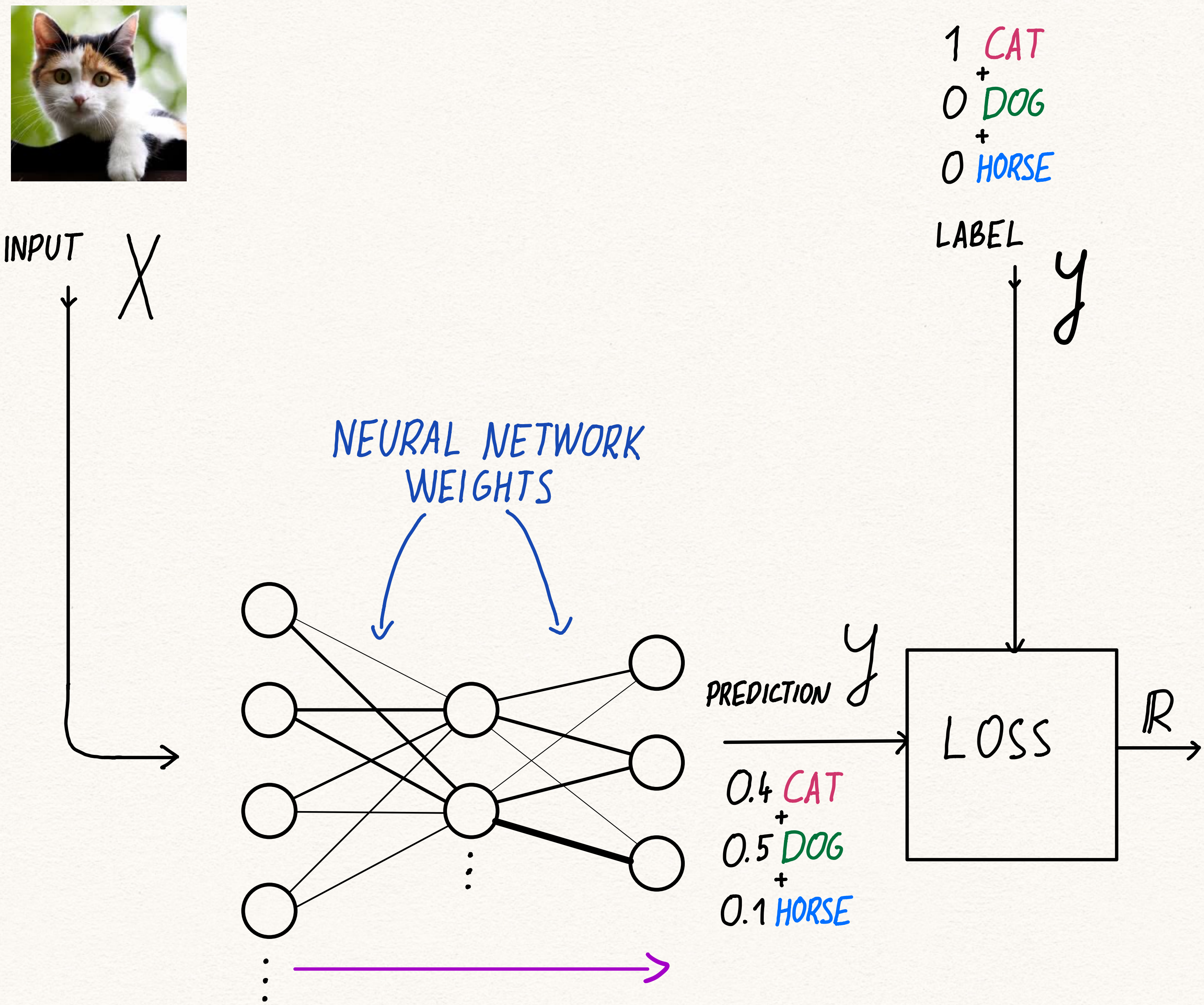
—

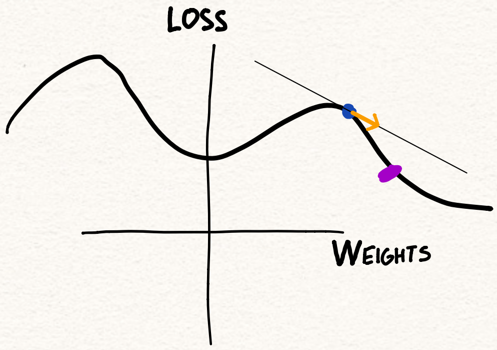
FOR DEEP LEARNING

SUPERVISED LEARNING WITH NEURAL NETWORKS
IN ONE SLIDE:

$$X \longrightarrow \boxed{\;?\;} \longrightarrow y$$

DATASET : List $X \times Y$





GRADIENT DESCENT ~ „OPTIMIZER"

- NN IS COMPUTATION PARAMETERIZED BY WEIGHTS
- BACKPROPAGATION OF CHANGES
- PARAMETER UPDATE – „OPTIMIZERS"

THIS SIMPLE STORY PERMEATES DEEP LEARNING!

PLAN FOR TODAY?

TAKE A BIRD'S EYE VIEW OF NEURAL NETWORKS

- TRACE OUT THE INFORMATION FLOW ABOVE
- PRECISELY WRITE DOWN ALL THE HIGH-LEVEL NOTIONS IN ISOLATION:
  - DIFFERENTIATION — REVERSE DERIVATIVE CATS.
  - BIDIRECTIONALITY — OPTICS/LENSES
  - PARAMETERIZATION — PARA

AND STUDY THEIR INTERACTION.

PARAMETERIZED OPTICS

AS A COMMON STRUCTURE BEHIND

- NEURAL NETWORKS
- LOSS FUNCTIONS
- OPTIMIZERS

- PAUL: CONCRETE EXAMPLES OF NEURAL NETWORKS

DIFFERENTIATION

- CARTESIAN (FORWARD) DIFFERENTIAL CATEGORIES (Blute et.al.)
- CARTESIAN REVERSE DIFFERENTIAL CATEGORIES (CRDC) (Cockett et.al.)

DEFINITION.

A CRDC $\mathcal{C}$ is a Cartesian left-additive category which for every map

$$f : A \longrightarrow B$$

has a REVERSE DIFFERENTIAL COMBINATOR

$$R[f] : A \times B \longrightarrow A$$

(compare $D[f] : A \times A \longrightarrow B$)

subject to 7 axioms.

EXAMPLE. Smooth is a CRDC. $\mathrm{Poly}_{\mathbb{Z}_2}$ IS A RDC

EXAMPLE. Let $\mathbb{R}^2 \xrightarrow{f} \mathbb{R}$ in Smooth.

$$(x,y) \longmapsto x^2 + 3yx$$

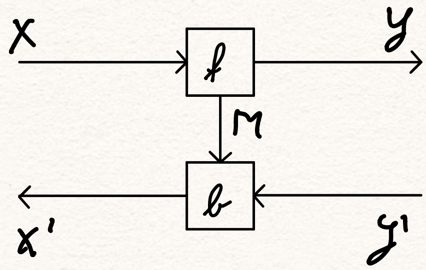
Then $R[f] : \mathbb{R}^2 \times \mathbb{R} \longrightarrow \mathbb{R}^2$

$$(x,y), w \longmapsto (2xw, 3xw)$$

PLAN: STUDY CRDC'S THROUGH OPTICS/LENSES

OPTICS / LENSES

DEFINITION. Let $\mathcal{C}$ be a SMC. Category $\mathrm{Optic}(\mathcal{C})$:



- Objects – pairs of objects $\binom{X}{X'}$ in $\mathcal{C}$

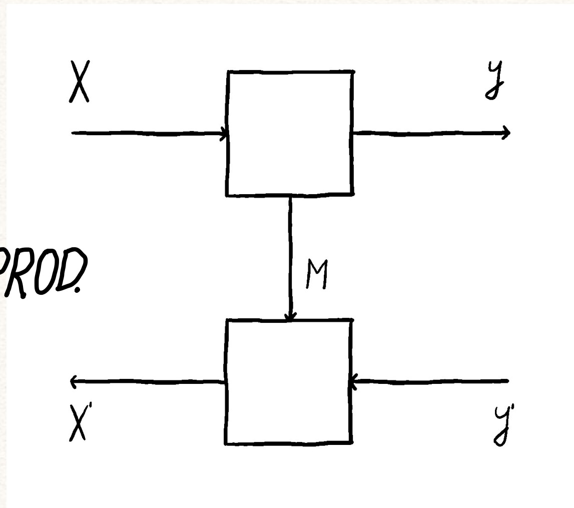
COEND

- $$\mathrm{Optic}(\mathcal{C})\left(\binom{X}{X'}, \binom{Y}{Y'}\right) = \int^{M:\mathcal{C}} \mathcal{C}(X, Y\otimes M) \times \mathcal{C}(Y'\otimes M, X')$$

(M, f, b) $\quad f : X \longrightarrow Y \otimes M$

$b : Y' \otimes M \longrightarrow X'$

PROP. If $\mathcal{C}$ is Cartesian,



$$\int^{M:\mathcal{C}} \mathcal{C}(X, M\times Y) \times \mathcal{C}(M\times Y', X')$$

$\cong$ UNIV. PROPERTY OF PROD.

$$\int^{M:\mathcal{C}} \mathcal{C}(X,Y) \times \mathcal{C}(X,M) \times \mathcal{C}(M\times Y', X')$$

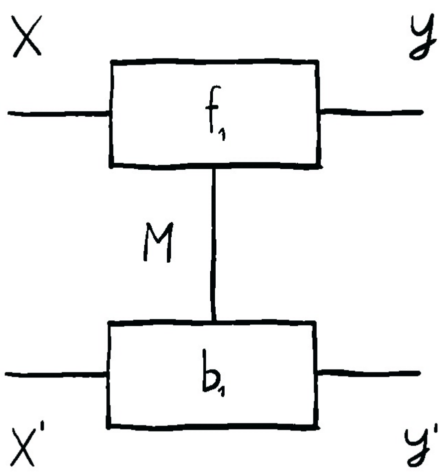
$\cong$ YONEDA REDUCTION

$$\int \underbrace{\mathcal{C}(X,Y)}_{get} \times \underbrace{\mathcal{C}(X\times Y', X')}_{put}$$

then $\mathrm{Optic}(\mathcal{C}) \cong \mathrm{Lens}(\mathcal{C})$

BIDIRECTIONAL INFORMATION FLOW

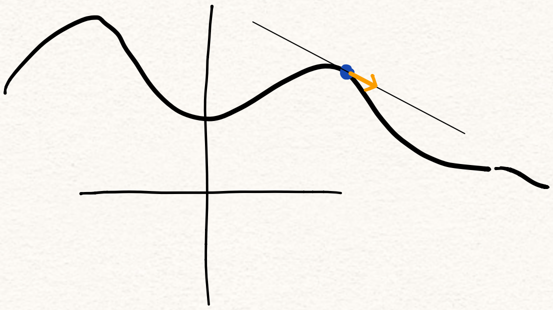
OPTICS CAN BE COMPOSED



PROPOSITION. $\mathbf{Optic}(\mathcal{C})$ is symmetric monoidal.

EXAMPLE.

GRADIENT DESCENT



$$P \times P' \xrightarrow{u} P$$
$$(p, \nabla p) \longmapsto p - \alpha \nabla p$$

is a lens, for $\mathcal{C} := \mathrm{Smooth}$

$$\begin{pmatrix} P \\ P \end{pmatrix} \xrightarrow{(id_P, u)} \begin{pmatrix} P \\ P' \end{pmatrix}$$

EXAMPLE. STATEFUL OPTIMIZERS

$$\begin{pmatrix} S \times P \\ S \times P \end{pmatrix} \longrightarrow \begin{pmatrix} P \\ P' \end{pmatrix}$$

- MOMENTUM,

  $get : P \times P \longrightarrow P$
  $(v, p) \longmapsto p$

  $put : P \times P \times P \longrightarrow P \times P$
  $(v, p, \nabla p) \longmapsto (v', p - v')$
  where $v' = \gamma v + \varepsilon p'$

- NESTEROV MOMENTUM

  $get : P \times P \longrightarrow P$
  $(v, p) \longmapsto p - \gamma v$

  put - same as above

- ADAGRAD
- ADAM

...

BACK TO CRDC's:

$f: A \longrightarrow B$ $\sim$ 'get' MAP OF A LENS

$R[f]: A \times B \longrightarrow A$ $\sim$ 'put' MAP OF A LENS

PROPOSITION.

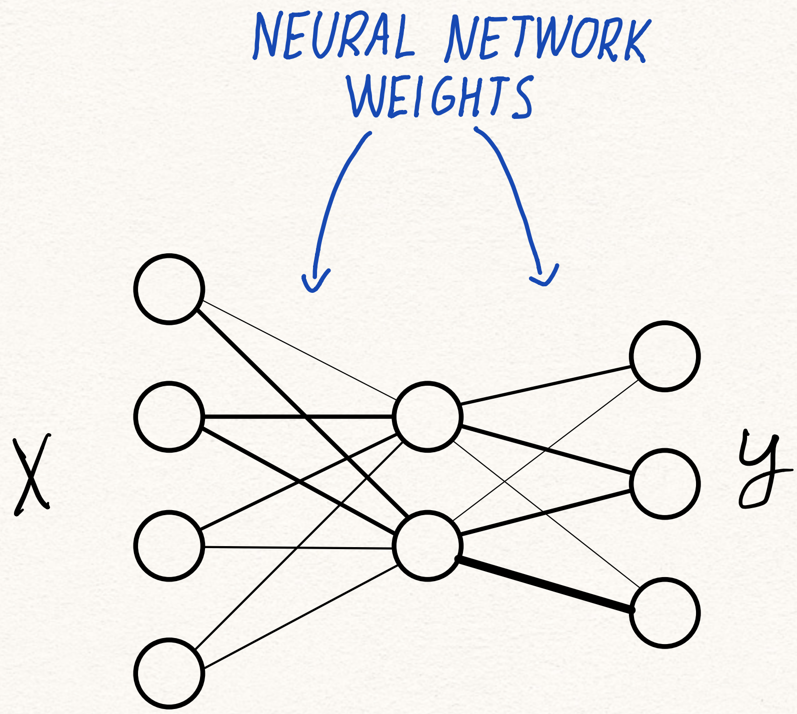
For each CRDC $\mathcal{C}$ there is a symmetric monoidal functor

$$\mathcal{C} \xrightarrow{F} \mathrm{Lens}(\mathcal{C}) \cong \mathrm{Optic}(\mathcal{C})$$

$$\begin{tikzcd}
A \arrow[r, mapsto] \arrow[d, "f"'] & (A, A) \arrow[d, "{(f, R[f])}"] \\
B \arrow[r, mapsto] & (B, B)
\end{tikzcd}$$

- THIS IS OUR FRAMEWORK FOR BACKPROPAGATION

PARAMETERIZATION



Fix a SMC $(\mathcal{C}, \otimes, I)$.

DEF. Bicategory $\mathbf{Para}(\mathcal{C})$

Objects - objects of $\mathcal{C}$

CATEGORY OF ELEMENTS

$$\mathbf{Para}(\mathcal{C})(A,B) = \int_{P:\mathcal{C}}^{\mathrm{op}} \mathcal{C}(P\otimes A, B)$$

$$A \xrightarrow{(P:\mathcal{C},\; f:P\otimes A \to B)} B$$

2-cells are reparameterizations: a 2-cell

$$\begin{tikzcd} A \arrow[r, bend left, "{(P,f)}"] \arrow[r, bend right, "{(Q,g)}"'] \arrow[r, phantom, "\Downarrow r"] & B \end{tikzcd}$$

is a map $Q \xrightarrow{r} P$ such that

$$\begin{tikzcd} Q\otimes A \arrow[rr, "r\otimes A"] \arrow[dr, "g"'] & & P\otimes A \arrow[dl, "f"] \\ & B & \end{tikzcd}$$

EXAMPLE.

$(\mathbf{Set}, \times, 1)$ SETS AND

$\mathbf{Para}(\mathbf{Set})$ PARAMETERIZED FUNCTIONS

$(\mathbf{Smooth}, \times, 1)$ EUCLIDEAN SPACES AND

$\mathbf{Para}(\mathbf{Smooth})$ PARAMETERIZED SMOOTH FUNCTIONS

$(\mathbf{Optic}(\mathcal{C}), \otimes, 1)$ PAIRS OF OBJECTS AND

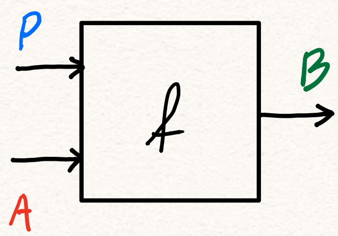
$\mathbf{Para}(\mathbf{Optic}(\mathcal{C}))$ PARAMETERIZED OPTICS

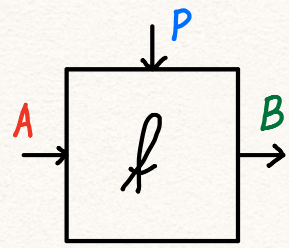
. . .

GRAPHICAL LANGUAGE

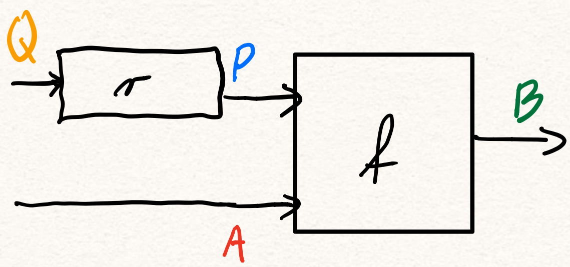
TEXTUAL NOTATION	STANDARD STRING DIAGRAM	2D STRING DIAGRAM

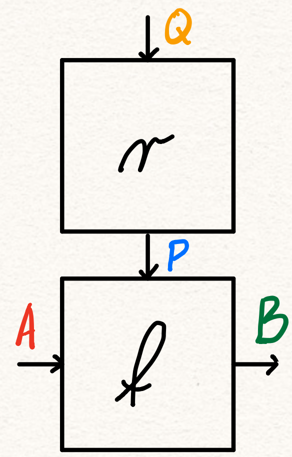
$f : P \otimes A \longrightarrow B$



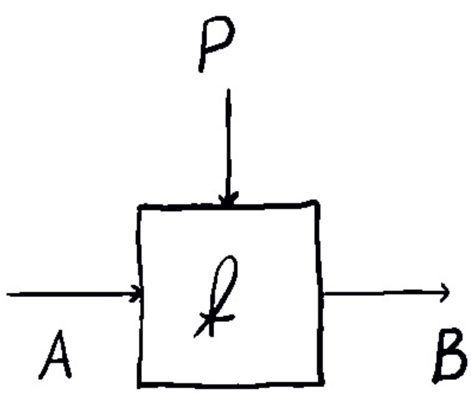


$$\begin{tikzcd} A \arrow[rr, bend left, "{(P,f)}"] \arrow[rr, bend right, "{(Q,g)}"'] \arrow[rr, phantom, "\Downarrow r"] & & B \end{tikzcd}$$

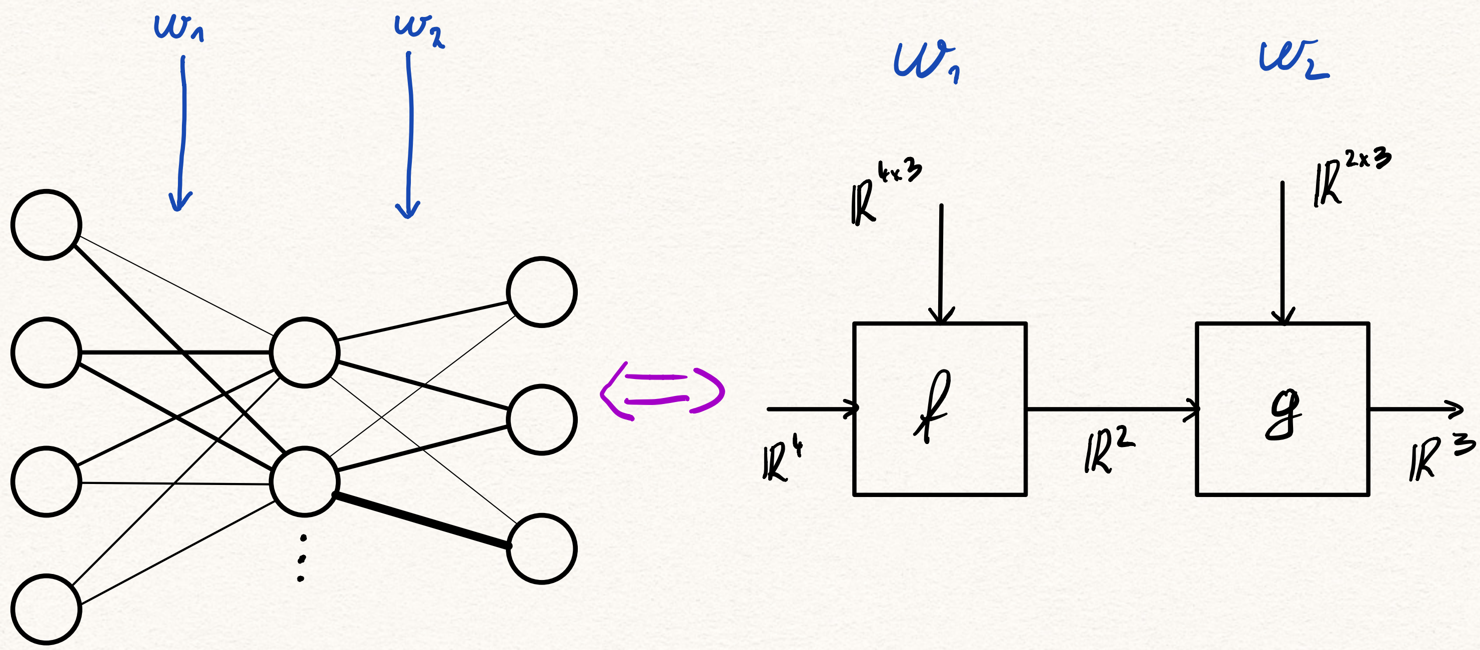




HOW DOES COMPOSITION WORK?



RECAP



Para IS NATURAL WITH RESPECT TO BASE CHANGE.

DEFINITION.

Let $G : \mathcal{C} \longrightarrow \mathcal{D}$ be a symm. monoidal functor. We define

$$
\begin{tikzcd}
\mathbf{Para}(G) : \mathbf{Para}(\mathcal{C}) \arrow[r] & \mathbf{Para}(\mathcal{D}) \\[-2em]
A \arrow[r, mapsto] \arrow[d, "{(P,f)}"'] & GA \arrow[d, "{(GP, f')}"] \\
B \arrow[r, mapsto] & GB
\end{tikzcd}
$$

where f' is the composite

$$G(P) \otimes G(A) \qquad\qquad\qquad\qquad G(B)$$

\+ MORE.

Para IS RICH IN CATEGORICAL STRUCTURE.

- CoKleisli category of a graded comonad
- Double category
- Actegorical Para

...

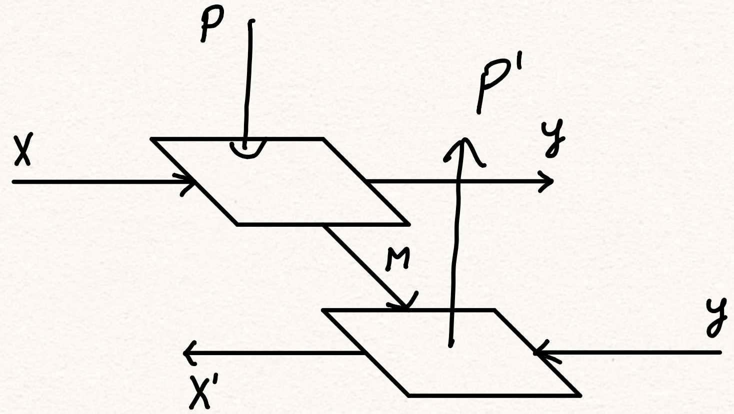
PARAMETERIZED OPTICS

$$\mathcal{C} \longmapsto \mathrm{Optic}(\mathcal{C}) \longmapsto \mathrm{Para}(\mathrm{Optic}(\mathcal{C}))$$

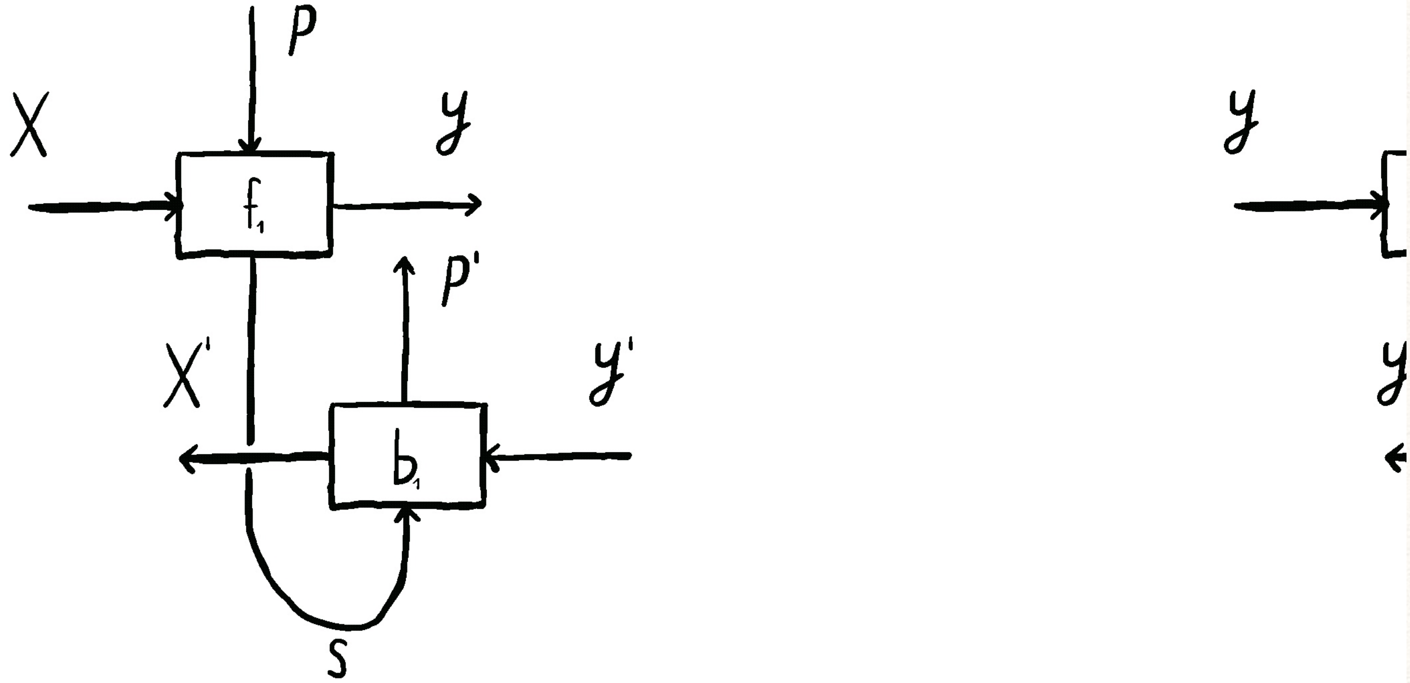
- Objects - objects of $\mathrm{Optic}(\mathcal{C})$ - pairs $\binom{X}{X'}$ in $\mathcal{C}$

- Morphisms $\binom{X}{X'} \xrightarrow{\left(\binom{P}{P'},\, f\right)} \binom{Y}{Y'}$ where $f : \binom{P \otimes X}{P' \otimes X'} \longrightarrow \binom{Y}{Y'}$

  (M, f, b)



- WE CAN COMPOSE PARAMETERIZED OPTICS



- We automatically get two parameter ports

- A 2-cell
$$\begin{tikzcd} \begin{pmatrix} X \\ S \end{pmatrix} \arrow[rr, bend left, "{\left(\begin{smallmatrix} P \\ Q \end{smallmatrix}, f\right)}"] \arrow[rr, bend right, "{\left(\begin{smallmatrix} Z \\ W \end{smallmatrix}, g\right)}"'] \arrow[rr, phantom, "\Downarrow r"] & & \begin{pmatrix} Y \\ R \end{pmatrix} \end{tikzcd}$$
is an optic

$$\begin{pmatrix} Z \\ W \end{pmatrix} \xrightarrow{r} \begin{pmatrix} P \\ Q \end{pmatrix}$$

THEOREM.

GRADIENT DESCENT IS A 2-cell IN $\mathrm{Para}(\mathrm{Optic}(\mathcal{C}))$.

(Since it is a lens)

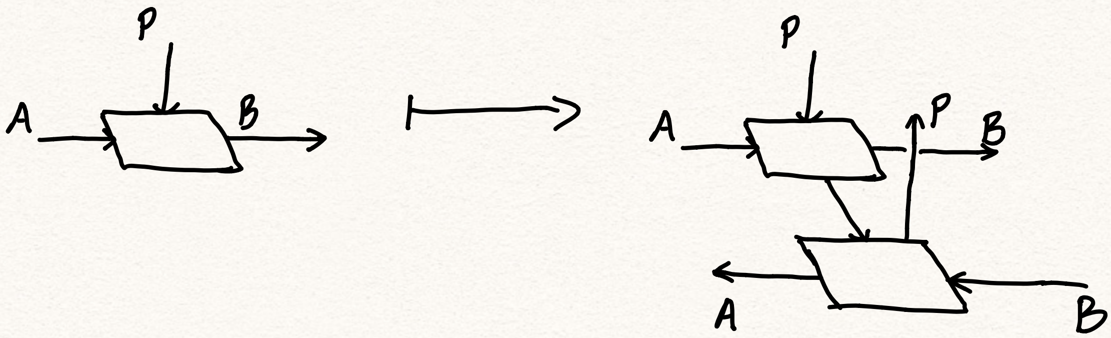
THEOREM

APPLYING Para TO THE CRDC FUNCTOR

$$\mathcal{C} \xrightarrow{F} \mathbf{Optic}(\mathcal{C})$$

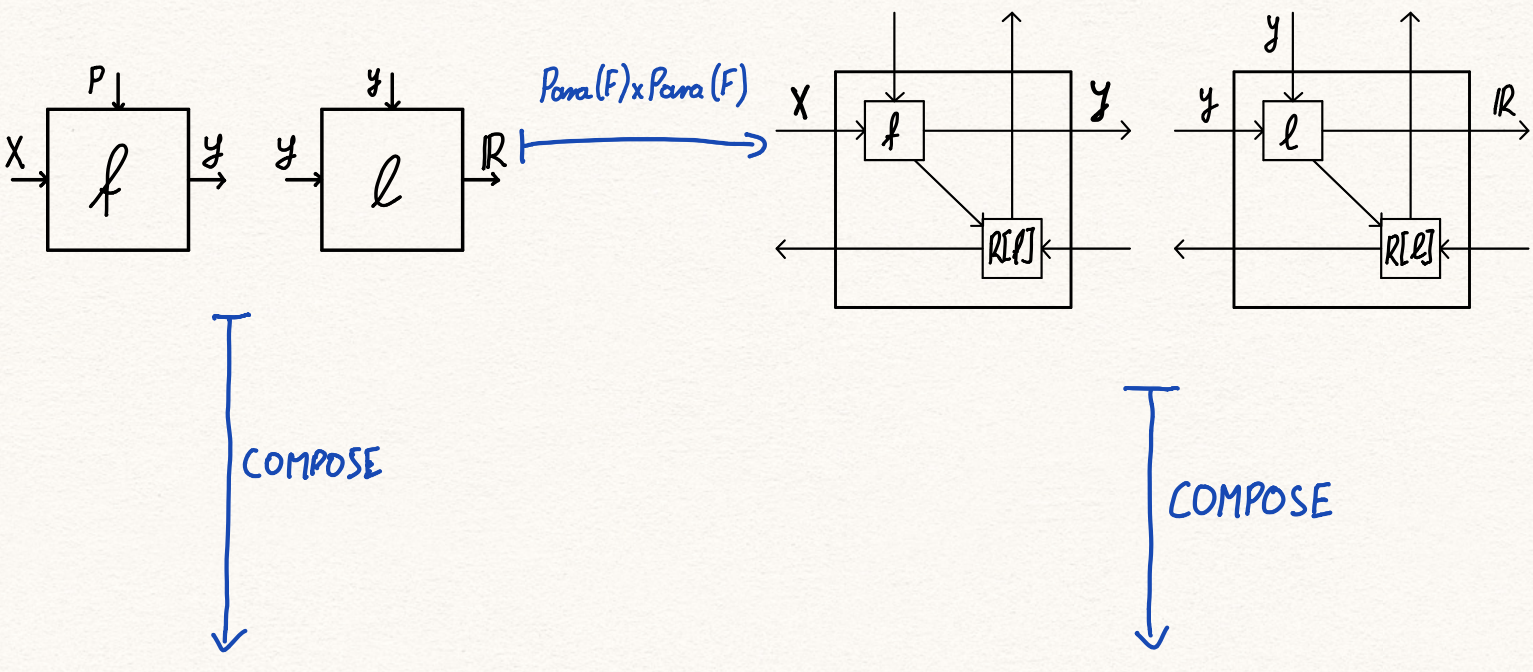
RESULTS IN A FUNCTOR

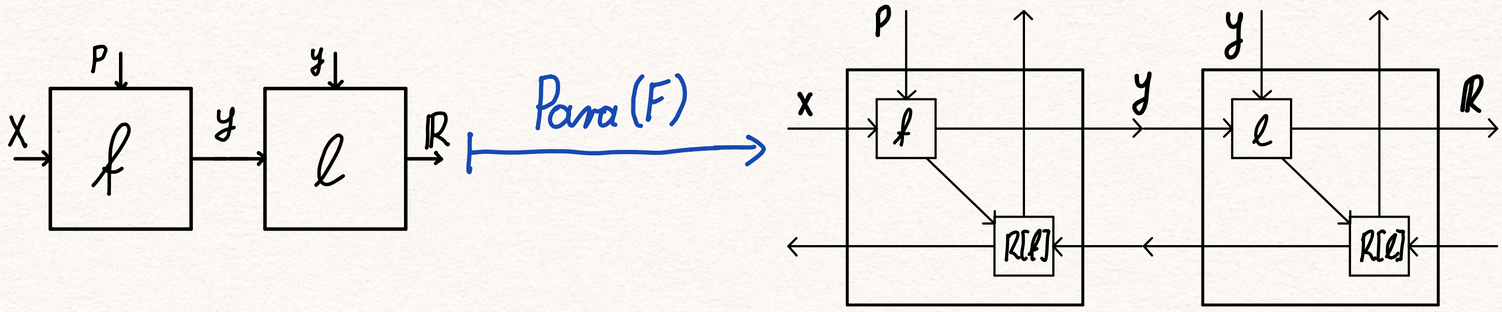
$$\mathbf{Para}(\mathcal{C}) \xrightarrow{\mathbf{Para}(F)} \mathbf{Para}(\mathbf{Optic}(\mathcal{C}))$$



• FUNCTORIALITY IS IMPORTANT!

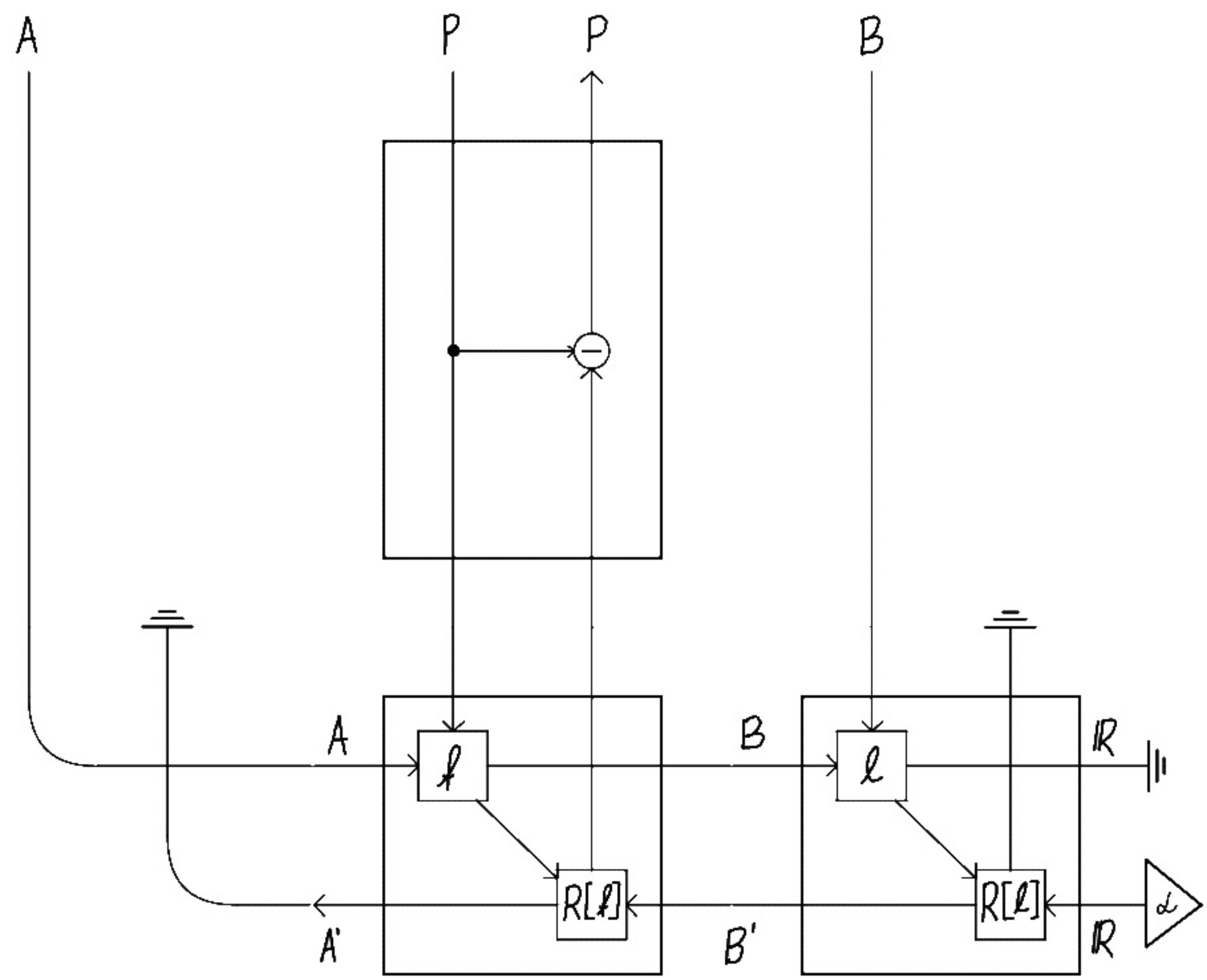
EXAMPLE. A NEURAL NETWORK + A LOSS FUNCTION





WE CAN PUT THE PIECES TOGETHER.

SUPERVISED LEARNING

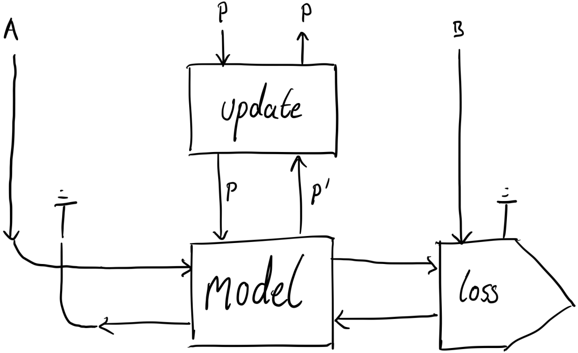


Categorical Foundations of Gradient-Based Learning

Recap

So far[1]:

- Para and Lens
- Optimizers, loss functions, models all (parametrised) lenses
- Putting them together, we get this:



Now: make these boxes more transparent...

[1] Cruttwell et al., "Categorical Foundations of Gradient-Based Learning."

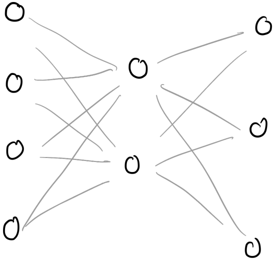
Next up

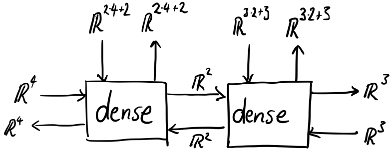
- Theme: **How to Build a Neural Network out of Lenses**
- Choosing the model is a creative process
- For an example problem, we'll look at the structure of one choice of model

Two goals:

- Show how to build a simple neural network out of lenses
- How to replace "classical" picture of neural networks with string diagrams

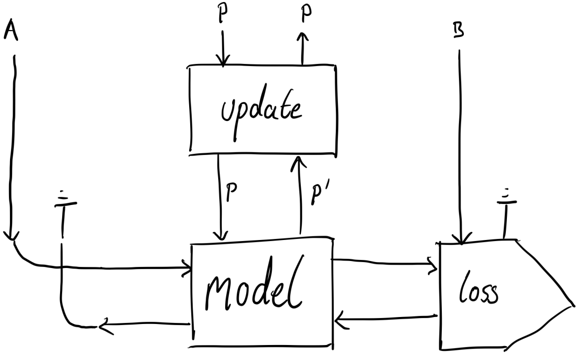
To String Diagrams





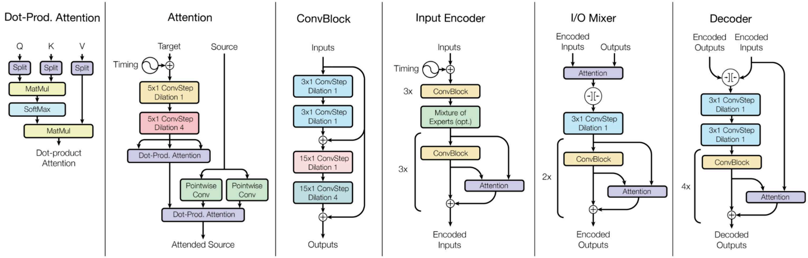
Three Levels of Detail 1: Learning

- The most "zoomed-out" view is the learner
- We look at the model as a kind of black box



Three Levels of Detail 2: Model Architecture

- The high-level structure of the model as a composition of “layers”
- Think of layers[2] as subroutines
- DL literature already starting to look string-diagrammatic[3]

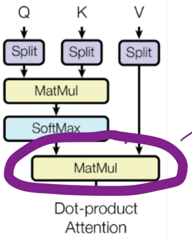


[2] Ambiguous terminology warning: “Layer” conflates objects and morphisms

[3] Kaiser et al., “One Model to Learn Them All.”

Three Levels of Detail 3: Layer

Finally: what are the pair of maps in our base category that make up a layer?



what's in here?

This Section of the Talk

- Supervised Learning & Reverse derivatives[4]
- End-To-End Example of a Neural Network
- Other Layer Examples
    - Weight Tying
    - Convolutional Layers
- Other settings (Circuits and $\mathsf{POLY}_{\mathbf{Z}_2}$)

[4] Cockett et al., "Reverse Derivative Categories."

Supervised Learning

In supervised learning, we want to learn a map

$$f : A \to B$$

from a dataset of examples

$$(a, b) \in A \times B$$

Now, based on our beliefs about the structure of A and B, we design a *parametrised* map:

$$\text{model} : P \times A \to B$$

and we search for some $\theta \in P$ such that $\text{model}(\theta, -)$ best represents the data.

Gradient-Based Learning

We want to use a datapoint $(a, b) \in A \times B$ to improve θ, so we need a map

$$??? : P \times A \times B \to P$$

The reverse derivative is almost what we want. For a map $f : A \to B$,

$$R[f] : A \times B' \to A'$$

(while in an RDC $A' = A$ and $B' = B$, it's useful think of the "primed" objects as representing **changes**)

So the reverse derivative of our model morphism has the following type:

$$R[\mathsf{model}] : P \times A \times B' \to P' \times A'$$

Updates, “Displacement” and Reverse Derivatives

This is not quite enough: we have two problems:

1. We have a “true” value $b \in B$ and a “predicted” value $\mathsf{model}(\theta, a) \in B$ but we need a B'
2. The reverse derivative gives us a P' and we want a P

This is exactly what the update and loss lenses are for:

$$\mathsf{update}_{\mathsf{put}} : P \times P' \to P$$

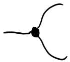
$$\mathsf{loss}_{\mathsf{put}} : B \times B \to B' \times B'$$

$$R[\mathsf{model}] : P \times A \times B' \to P' \times A'$$

Reverse Derivatives, Graphically

CARTESIAN STRUCTURE

copy



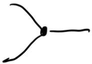
$x \mapsto \langle x, x \rangle$

discard

$x \mapsto \langle \rangle$

LEFT-ADDITIVE STRUCTURE

add

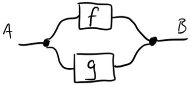


$x_1, x_2 \mapsto \langle x_1 + x_2 \rangle$

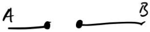
zero

$\square \mapsto \langle 0 \rangle$

Addition & zero maps



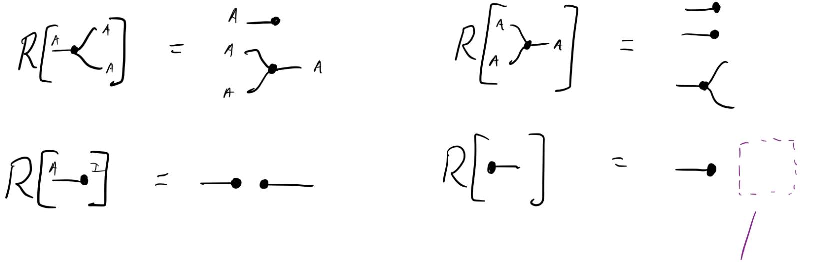
$f + g : A \to B$



$0 : A \to B$

Reverse Derivatives, Graphically

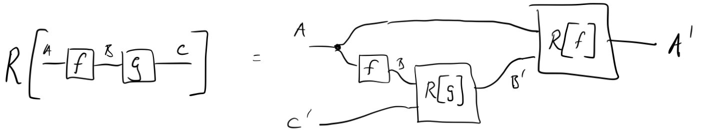
$$A \xrightarrow{f} B \quad \Longrightarrow \quad A \times B' \xrightarrow{R[f]} A'$$



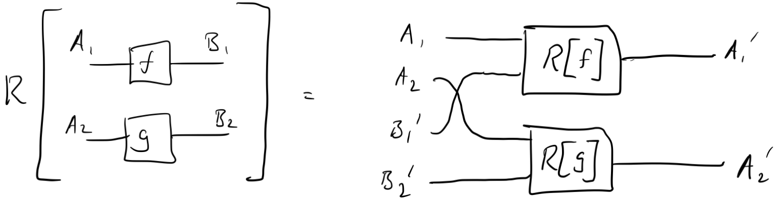
No input, so no CHANGE in input

Reverse Derivatives, Graphically

$$A \xrightarrow{f} B \quad \Longrightarrow \quad A \times B' \xrightarrow{R[f]} A'$$

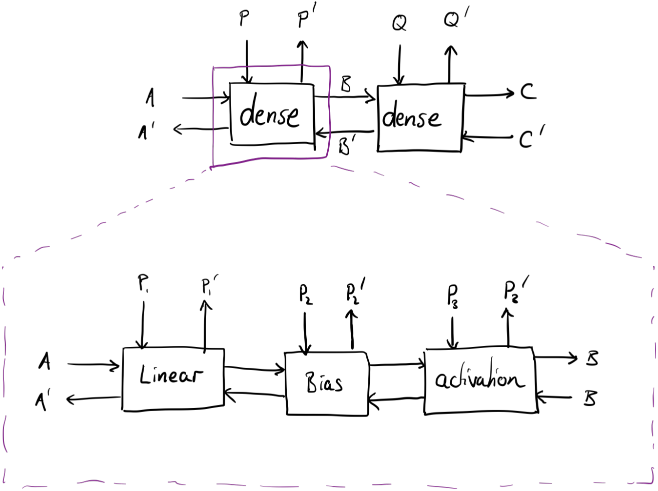


(RD.5

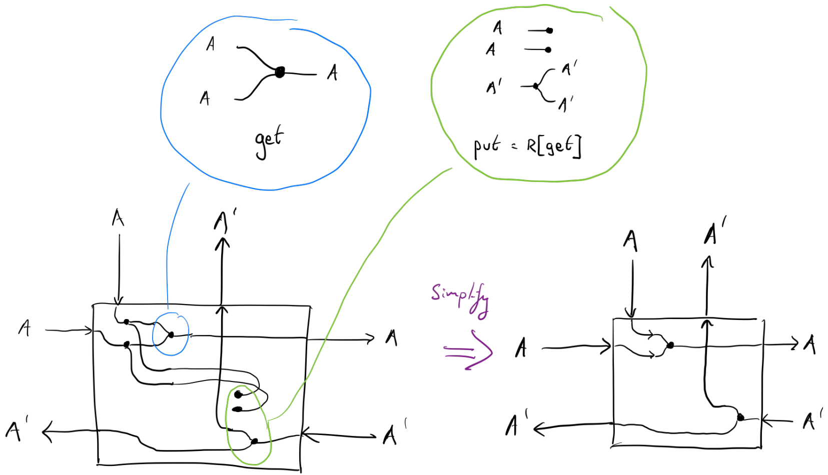


Neural Networks 1: Dense Layers

Now let's unpack a dense layer...



Neural Networks 2: Bias "Layer"



Neural Networks 3: ‘Linear’ Layer

- Parameters $P = \mathbb{R}^{b \cdot a}$ are the coefficients of a matrix
- Input $A = \mathbb{R}^a$ is an a-dimensional vector
- Forward pass multiplies the matrix by the vector:

$$\mathsf{get}(M, x) \mapsto Mx$$

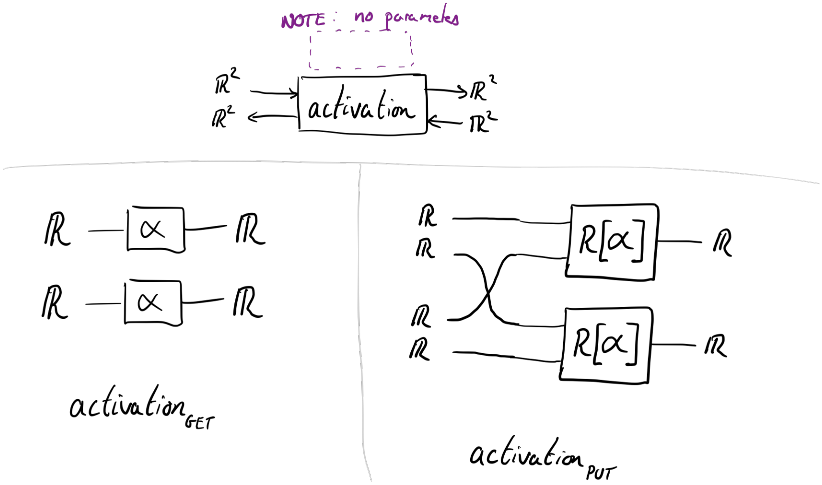
- Reverse pass does the “obvious” thing that typechecks: if we think of the get map as having the type

$$\mathsf{get} : \mathsf{Mat}(A, B) \times \mathsf{Vec}(A) \to \mathsf{Vec}(B)$$

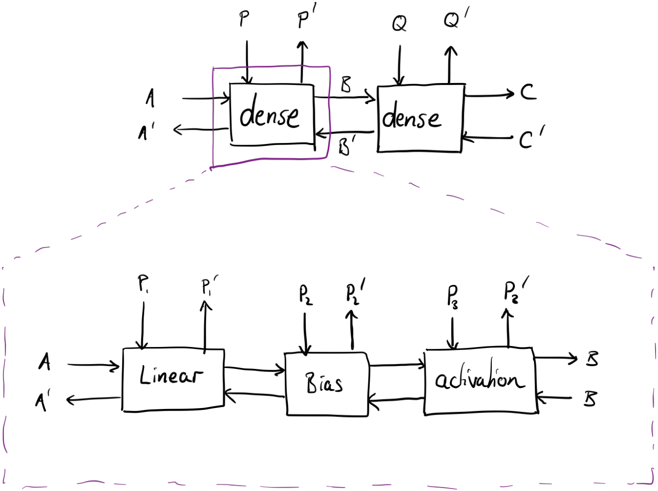
Then the codomain of the put map should be $\mathsf{Mat}(A, B) \times \mathsf{Vec}(A)$:

$$\mathsf{put}(M, x, y) \mapsto \langle y \otimes x, M^T y \rangle$$

Neural Networks 4: Activation Layer

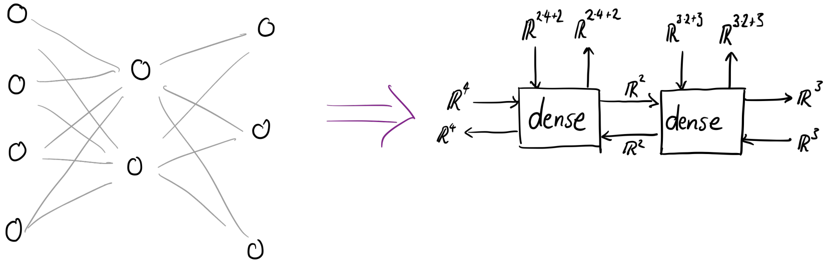


Neural Networks 5: Dense Layers (again)

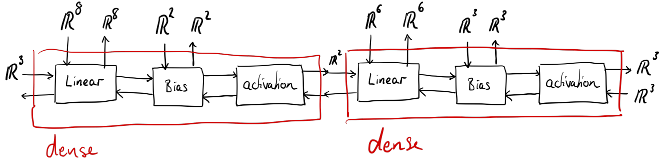


Neural Networks 6: Hidden Layer Neural Network

Returning to the "standard" picture of a neural network:

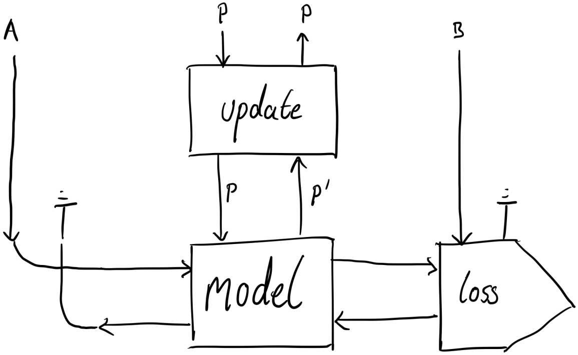


Expanding out "dense":

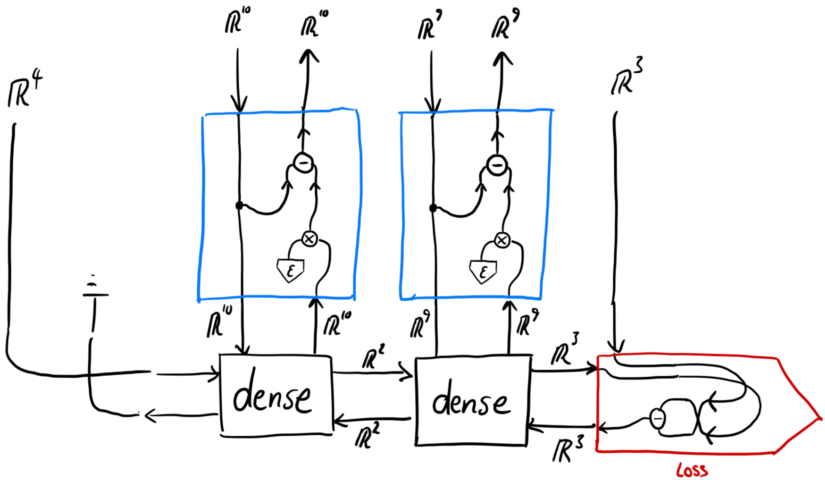


Update & Loss

Now let's substitute all parts into the full picture



Full Picture (Again)



Code

- Code implementing these ideas can be found here: https://github.com/statusfailed/numeric-optics-python/
- Includes this hidden layer neural network model
- Also includes a convolutional model for the MNIST dataset (more on this shortly...)

More

- Other Layer Examples
  - Weight Tying
  - Convolutional Layers
- Other settings (Circuits and $\mathrm{POLY}_{\mathbf{Z}_2}$)

"Weight Tying"

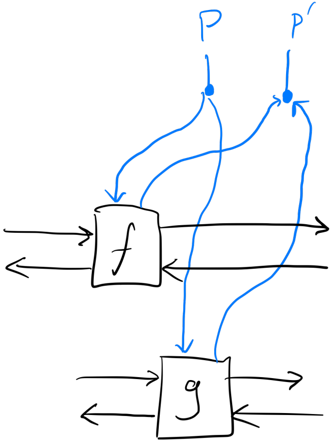
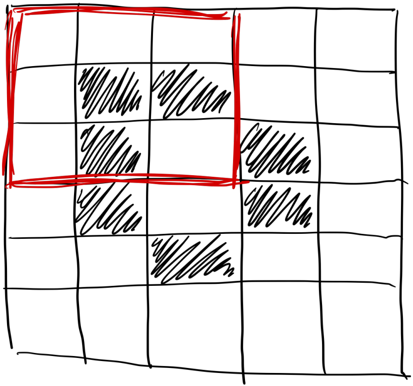


Image Processing

- Example problem: image processing, e.g. digit recognition
- Convolution layer: features with spatial locality

Convolutional Layers



Other Settings: $\mathsf{POLY}_{\mathbf{Z}_2}$

- $\mathsf{POLY}_{\mathbf{Z}_2}$ is an RDC
- We can still think of morphisms as functions
- Gradient-based learning still works[5]
- Strange possibilities for layers: the LUT

[5] Wilson and Zanasi, "Reverse Derivative Ascent."

References

Cockett, Robin, Geoffrey Cruttwell, Jonathan Gallagher, Jean-Simon Pacaud Lemay, Benjamin MacAdam, Gordon Plotkin, and Dorette Pronk. "Reverse Derivative Categories," 2019. http://arxiv.org/abs/1910.07065.

Cruttwell, G. S. H., Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. "Categorical Foundations of Gradient-Based Learning," 2021. http://arxiv.org/abs/2103.01931.

Kaiser, Lukasz, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. "One Model to Learn Them All," 2017. http://arxiv.org/abs/1706.05137.

Wilson, Paul, and Fabio Zanasi. "Reverse Derivative Ascent: A Categorical Approach to Learning Boolean Circuits." *Electronic Proceedings in Theoretical Computer Science* 333 (February 2021): 247–60. https://doi.org/10.4204/eptcs.333.17.