Categories of Neural Networks

1st Michael Moy
Department of Mathematics
Colorado State University
Colorado, USA
michael.moy@colostate.edu

2nd Robert Cardona
Department of Mathematics
University at Albany
New York, USA
rlcardona@albany.edu

3rd Alan Hylton
Near Space Network
NASA Goddard
Maryland, USA
alan.g.hylton@nasa.gov

***Abstract*—We introduce new categories of neural networks that we hope will help us expose and explore their underlying structure and relationships. Our categories represent neural networks as objects and present structure for comparing neural networks with the same number of layers. In this paper we discuss the construction, properties, and currently known limitations – all with specific, illustrative examples. There are a number of directions this work can move towards, and we conclude with a section on future work summarizing several of these directions.**

I. Introduction

In this paper, we propose an approach to forming a category of neural networks. This has been accomplished in [1] – however, their approach was more focused on supervised learning than the neural networks themselves. In [2], the authors make categories out of given neural networks. Our goal is to better understand the space of neural networks, rather than algorithms that act on them, hopefully leading to a means to "look under the hood." Indeed, while neural networks have become increasingly popular because of their ability to perform complicated tasks, there is still a limited understanding of how a neural network performs its task once it is trained. One example comes from the form of *adversarial input*, where neural networks can be forced to consistently misclassify inputs with high confidence by applying small perturbations, which is argued to stem from their linear nature [3]. Along with this, there is very little ability to compare different networks, other than by comparing the performance of two networks trained to perform the same task. We propose a method of mathematically comparing neural networks without reference to the tasks they have been trained to perform or the training processes. In this setting, a neural network is nothing more than a sequence of functions between its layers.

A more specific goal is to create a category that, in some sense, *maximizes categorical richness*. Background information on category theory can be found, for instance, in [4], [5]. To begin, we may consider what limits and colimits exist in a category. In [1], the category **NNet** has as objects $\mathbb{N} = \{1, 2, \ldots\}$, $\mathbf{NNet}(n, m) = \{\text{networks of type } (n, m)\}$, and composition is concatenation where possible. The natural candidate for an initial object is 1, but there is no unique arrow from 1 (or any object) to any other object. Then **NNet** does not have an initial object, and similarly, there is no terminal object. In the discussion that follows, we work towards our desired categories, which we will show have terminal objects and binary products.

As we do not consider the training process of neural networks, we will focus on the functions that make up the layers – we refer those looking for a guide on neural networks to [6]. We will discuss neural networks in two ways: as sequences of layers of nodes with connections between them, as they are often presented, and as sequences of functions between Euclidean spaces, which is a concise, equivalent description. The common terminology of weights and biases will be used when discussing individual nodes and their connections to other nodes. The functions between layers built from such connections are affine functions between Euclidean spaces, that is, functions $f : \mathbb{R}^n \to \mathbb{R}^m$ of the form $f(x) = Ax + b$ where A is a matrix and b is a vector. An affine subspace of $\mathbb{R}^n$ is the set of solutions to an equation of the form $Ax = b$. Affine functions and affine subspaces will both play important roles.

Activation functions are a critical part of neural networks. Activation functions are often viewed as being applied to each individual node; we will also use the term "activation function" to mean either a function applied to an individual node or a function applied to an entire layer. It is worth noting that some activation functions used in practice cannot be expressed in terms of functions on individual nodes, but rather require the entire layer as an input: the softmax activation function is an example. With this terminology, a function between consecutive layers of a typical neural network can be concisely described as an affine function followed by an activation function. We will consider both networks with arbitrary activation functions and networks with the common activation function ReLU. ReLU, written here as $\sigma : \mathbb{R} \to \mathbb{R}$, is defined by $\sigma(x) = \max\{0, x\}$. We will also use σ to mean the function obtained by applying ReLU to each coordinate of a point in $\mathbb{R}^n$, for any n.

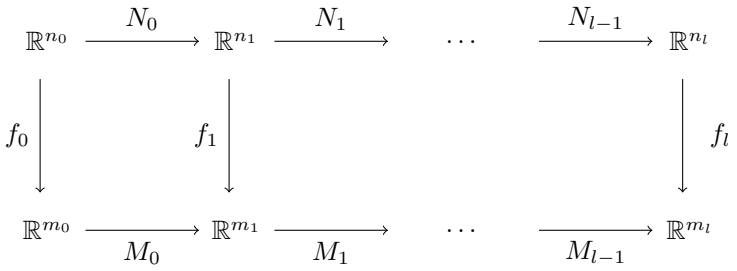
The goal of this paper is to construct and examine categories of neural networks. A very general category is constructed, with two specific types of subcategories being our primary focus. Defining this general category provides the opportunity for future work to focus on other subcategories; these will be different categories with similar constructions to those examined here. In each of these categories, the neural networks are the objects, and a morphism between two networks can be thought of as a layer by layer comparison of the networks.

As such, the categories will be limited to networks of a given length. We will examine several properties of our two main categories and provide multiple examples of morphisms. The final section contains ideas for future work.

II. A General Definition

In order to compare two neural networks of length l, we will consider cases in which we can "translate" between the corresponding layers of two networks. Mathematically, this will be accomplished by a sequence of functions between the corresponding layers. We thus arrive at a first, very general definition of a category of neural networks.

Define a neural network of length l to be a sequence of functions $N = (N_0, N_1, \ldots, N_{l-1})$ between Euclidean spaces $\mathbb{R}^{n_0}, \mathbb{R}^{n_1}, \ldots \mathbb{R}^{n_l}$, with $N_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ for each i. The functions N_i will be referred to as layer functions of N. If $N = (N_0, N_1, \ldots, N_{l-1})$ and $M = (M_0, M_1, \ldots, M_{l-1})$ are neural networks of length l, define a morphism $f : N \to M$ of neural networks of length l to be a sequence of functions $(f_0, f_1, \ldots, f_l)$ as in the following diagram such that each rectangle including f_0 commutes. That is, we require $f_k \circ N_{k-1} \circ \cdots \circ N_1 \circ N_0 = M_{k-1} \circ M_{k-2} \circ \cdots \circ M_0 \circ f_0$ for all $1 \leq k \leq l$.



Note that this condition is equivalent to requiring that any composite of functions in the diagram from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{m_i}$ is equal, for any i. This ensures that any point in $\mathbb{R}^{n_0}$ is sent to its images in M consistently by the f_i. We do not require that each square commute because data is never fed into an inner layer of a network, and thus we need not consider the points in $\mathbb{R}^{n_i}$ that are not in the image of the previous layer functions. However, note that if all squares commute, then we have a well defined morphism, even though the converse is not true. Composition of morphisms is defined layer by layer:

$$(f_0, f_1, \ldots, f_l) \circ (g_0, g_1, \ldots, g_l) = (f_0 \circ g_0, f_1 \circ g_1, \ldots, f_l \circ g_l).$$

With these definitions, the set of neural networks of length l form a category: associativity of composition follows from the associativity of function composition, and for any neural network, the sequence of appropriate identity functions is the identity morphism.

This definition is likely too general to be of much use by itself. Allowing a neural network to be made up of arbitrary functions does not reflect how neural networks are used in practice, and allowing the morphisms to be made up of arbitrary functions allows more flexibility than may be desirable. In order to work with more useful sets of networks and morphisms, we can impose various restrictions on the layer functions and the morphisms. For instance, a simple adjustment is to require that each layer function N_i in a network N be an affine function followed by any activation function. A stricter requirement is to specify that each be an affine function followed by a specific activation function. For morphisms, a simple requirement is that all f_i in a morphism f be continuous functions. A stricter requirement is that all f_i be linear functions. With any such adjustment to the definition, as long as the new class of morphisms is closed under composition and contains the identity morphisms, we obtain a subcategory of our original category.

We will continue to use the notation as shown above for networks, with a single capital letter (such as N) representing the entire network, the same capital letter with subscripts (such as N_1) indicating the layer functions of the network, and the lowercase letter with subscripts (such as n_1) indicating the dimensions of the Euclidean spaces. Morphisms will also follow the same notation as above.

III. More Specific Definitions

In practice, the layer functions of a network are affine functions followed by some activation function. We will only consider such networks from here on. The activation functions are the key feature of neural networks: it is a common observation that nonlinear activation functions are what allow a network to express complicated functions (really the important property is that the activation function be non-affine). With this in mind, we consider morphisms that are made up of affine maps. These form a general class of maps that agree with how neural networks are constructed but avoid introducing additional nonlinearity into the category. Such a morphism thus relates two networks by relating their use of their activation functions. Restricting to these morphisms will

lead to certain nice properties and has the additional benefit that all such morphisms are computationally practical.

Thus, we define a category **AffineNet**$_l$, the "category of length l neural networks with affine morphisms." A network N in this category will be as above, with each N_i required to be equal to an affine function followed by any activation function. The morphisms will be of the form $f = (f_0, f_1, \ldots, f_l)$ as above, where each f_i is required to be an affine function. These will simply be referred to as affine morphisms.

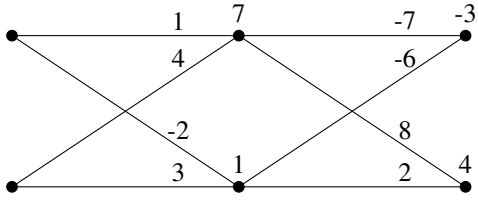
We now turn to even more specific requirements. We will form a category of ReLU neural networks, that is, networks N where each $N_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ is of the form $N_i = \sigma \circ A_i$ for some affine function $A_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$. Let **ReluAffineNet**$_l$ be the category of ReLU neural networks with affine morphisms.

While the notation above is a little cumbersome, it allows for flexibility in forming new, similar categories. For instance, one may wish to analogously define categories such as **LinearNet**$_l$ or **ReluLinearNet**$_l$ or **TanhAffineNet**$_l$.

In what follows, we will restrict our attention to these two categories, with a particular focus on **ReluAffineNet**$_l$. Working with ReLU networks and affine morphisms is especially appealing because ReLU interacts reasonably well with linear and affine functions: the relevant property is $\sigma(cx) = c\sigma(x)$ for nonnegative c and any real x.

For future work, we will note that it is possible to modify these categories to allow empty sets in place any $\mathbb{R}^{n_i}$. This has some merit, as the empty set can be considered as an affine subspace of a Euclidean space. These modified categories will be mentioned occasionally, as they have slightly different properties. Whether there is any real benefit to working with these modified categories is questionable, and networks with empty sets are certainly are not realistic.

Examples of networks and morphisms will appear later. To represent networks concisely and to help provide intuition, we will show networks as they are commonly drawn, as layers of nodes connected by edges. We will place weights above the edges and biases above the nodes, and state the activation functions when necessary. So for instance, we can depict a network with ReLU activation functions in the following equivalent ways.



$$\mathbb{R}^2 \xrightarrow{N_0} \mathbb{R}^2 \xrightarrow{N_1} \mathbb{R}^2$$

$$N_0(x) = \sigma\left(\begin{pmatrix} 1 & 4 \\ -2 & 3 \end{pmatrix} x + \begin{pmatrix} 7 \\ 1 \end{pmatrix}\right)$$

$$N_1(x) = \sigma\left(\begin{pmatrix} -7 & -6 \\ 8 & 2 \end{pmatrix} x + \begin{pmatrix} -3 \\ 4 \end{pmatrix}\right)$$

We will see that the definition of affine morphisms yields many desirable types of morphisms and that these categories contain some common category theoretic constructions, along with meaningful interpretations. Near the end, we will also see a nontrivial example of a morphism, one that is slightly more complicated than many of the simple examples considered throughout and that is between two networks without an immediately obvious relationship. Given the broad class of morphisms found in these categories, there is potential that this framework could lead to a better understanding of neural networks and the ability to compare networks on a deeper level.

IV. Constant Morphisms

Given any two networks N and M in either **AffineNet**$_l$ or **ReluAffineNet**$_l$, there always exists at least one morphism from N to M. We can construct a "constant morphism" $f = (f_0, f_1, \ldots, f_l)$, where each f_i is a constant function. For any $x_0 \in \mathbb{R}^{m_0}$, let $f_0 : \mathbb{R}^{n_0} \to \mathbb{R}^{m_0}$, be the constant function with value x_0, and for each i, let $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$ be the constant function with value $M_{i-1} \circ \cdots \circ M_1 \circ M_0(x_0)$. Note that we obtain distinct morphisms for distinct choices of x_0, and as long as $m_0 > 0$, we in fact have an infinite number of such morphisms. Also note that if one works with the modified categories that allow empty sets, there do not exist constant morphisms to networks that include empty sets. The existence of such constant morphisms is reassuring; they are similar to constant functions between topological spaces, or group homomorphisms that send all elements to the identity. They represent the most trivial way that one network can be mapped to another.

V. Isomorphisms

We have the following intuitive result on isomorphisms:

Theorem 1. *In both **AffineNet**$_l$ and **ReluAffineNet**$_l$, a morphism $(f_0, f_1, \ldots, f_l) : N \to M$ is an isomorphism if and only if each f_i is a bijection.*

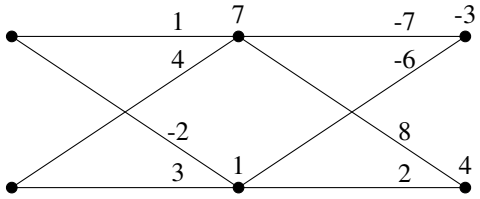
Proof. If $(f_0, f_1, \ldots, f_l)$ is an isomorphism, then the existence of a two-sided inverse shows each f_i is a bijection. Conversely, if each f_i is a bijection, it can be checked that $(f_0^{-1}, f_1^{-1}, \ldots, f_l^{-1})$ is a morphism from M to N. □

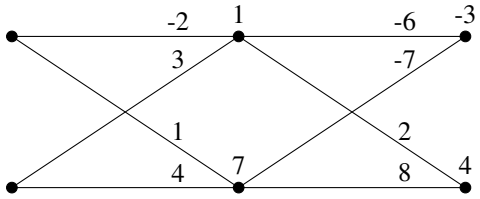
In particular, if a morphism $f : N \to M$ is an isomorphism, then we must have $n_i = m_i$ for each i. That is, two

isomorphic networks must have the same number of nodes in each layer. By this theorem, isomorphic neural networks N and M "behave the same," meaning that the composite functions $N_{i-1} \circ \cdots \circ N_0$ and $M_{i-1} \circ \cdots \circ M_0$ are the same up to invertible affine transformations.

Affine isomorphisms include some of the most simple transformations that one might expect to be isomorphisms. For instance, one can imagine beginning with a neural network with the same activation function for every node, and simply changing the order of the nodes in some layer, moving all weights and biases along with them. The resulting network should intuitively be isomorphic to the original, since it functions in exactly the same way, up to this rearrangement of nodes. There does in fact exist an affine isomorphism between such networks: the function f_i corresponding to the changed layer is given by a permutation matrix, and all other f_i are identities. Similarly, several layers may be rearranged at once to give another isomorphic network. These examples are found in both **AffineNet**$_l$ and **ReluAffineNet**$_l$.

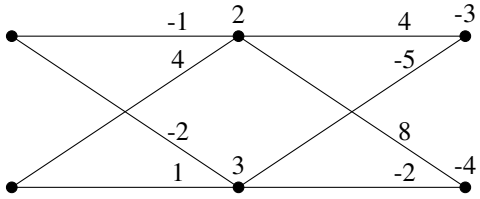
As an example, consider the following two networks, with the same activation functions on all nodes.

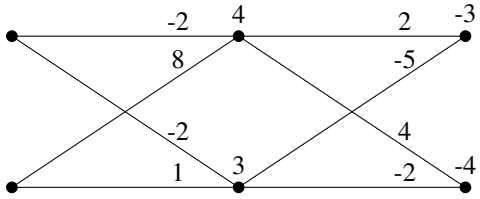




These two networks are the same, except for the flipped nodes in the middle layer. Thus, (id, f_1, id) with $f_1(x) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} x$ is an isomorphism (either from the first network to the second or vice versa). It is tedious but informative to check this and future examples in full.

Another simple type of isomorphism in **ReluAffineNet**$_l$ is given by scaling the value of an individual node by a positive scalar c, and simultaneously scaling all outgoing weights associated with that node by $\frac{1}{c}$. Because $\sigma(cx) = c\sigma(x)$ for positive c, this leaves all other nodes unchanged, so intuitively we expect this to be an isomorphism. Again, this is described by an affine isomorphism f, with the $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ at the layer in question simply scaling one component of $\mathbb{R}^{n_i}$ by c. An example is given in the following two networks with ReLU activations.





An isomorphism from the first network to the second is (id, f_1, id) with $f_1(x) = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} x$. Checking this relies on the property $\sigma(cx) = c\sigma(x)$ for nonnegative c and any real x

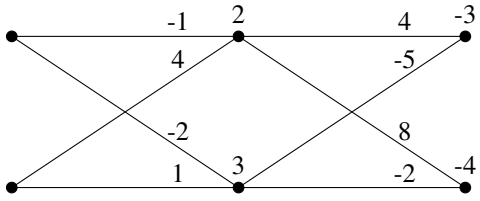
VI. Initial and Terminal Objects

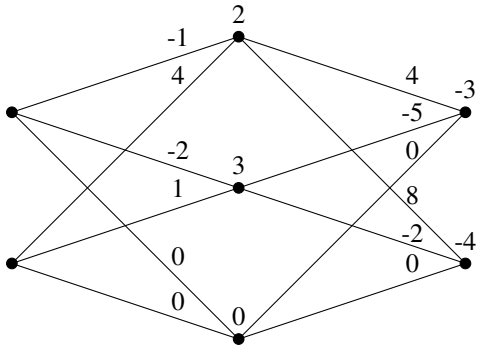
Both **ReluAffineNet**$_l$ and **AffineNet**$_l$ have a terminal object, which we will call the "terminal network," T, in which every layer function is the unique function from $\mathbb{R}^0$ to $\mathbb{R}^0$. A morphism from this terminal network to a network N can be thought of as tracing out a particular path in the network N, and thus $\mathrm{Hom}(T, N)$ represents the set of all paths that an input may take through the network N. Comparing to the description of constant morphisms, we see that any constant morphism factors through the terminal network. Investigating other simple hom-sets could make for interesting future work.

Neither category has an initial object: for any network N, there exist multiple constant morphisms $N \to M$ as long as $m_0 > 0$. The lack of an initial object shows that in general, colimits do not necessarily exist. However, if one chooses to modify the categories to allow empty sets, then the network consisting entirely of empty sets and empty functions is an initial object in either category.

VII. Split Monomorphisms and Epimorphisms

Next we will consider a specific type of split monomorphism and split epimorphism in **ReluAffineNet**$_l$ (similar morphisms for other networks with non-ReLU activations are also found in **AffineNet**$_l$, with some minor modifications). Given a network, we could imagine adding in an additional node at one layer, with all its weights and its bias set to zero, and with all weights connected to it in the following layer set to zero. We should expect some relationship to the original network, since the new node is completely inactive. However, the two networks cannot be isomorphic, as they have a different number of nodes in one layer. The networks pictured show an example.





Starting with a network N, we will add an inactive node to layer k to form the network M. We consider the case where layer k is an inner layer, that is, $1 \leq k \leq l-1$; similar reasoning can be applied for $k = 0$ and $k = l$. For all $i \neq k$, let $m_i = n_i$, and let $m_k = n_k + 1$, where the additional coordinate in $\mathbb{R}^{m_k}$ is our new node. For all i not equal to $k-1$ or k, let $M_i = N_i$. We proceed to define M_{k-1} and M_k. Define $j : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k+1}$ to be the embedding of $\mathbb{R}^{n_k}$ into the first n_k coordinates of $\mathbb{R}^{n_k+1}$, with the last coordinate constant at zero. Set $M_{k-1} = j \circ N_{k-1}$; this is equivalent to the description that the new node has weights and bias equal to zero, since the activation function is σ. Similarly, let $\pi : \mathbb{R}^{n_k+1} \to \mathbb{R}^{n_k}$ be the projection onto the first n_k components, and set $M_k = N_k \circ \pi$; this is equivalent to the description that weights in the following layer connected to the new node are zero. We have the following diagram.

$$\begin{array}{ccccc}
\mathbb{R}^{n_{k-1}} & \xrightarrow{N_{k-1}} & \mathbb{R}^{n_k} & \xrightarrow{N_k} & \mathbb{R}^{n_{k+1}} \\
id \downarrow \uparrow id & & j \downarrow \uparrow \pi & & id \downarrow \uparrow id \\
\mathbb{R}^{n_{k-1}} & \xrightarrow[M_{k-1}]{} & \mathbb{R}^{n_k+1} & \xrightarrow[M_k]{} & \mathbb{R}^{n_{k+1}}
\end{array}$$

By the definition of M_{k-1} and M_k, the respective squares commute. Next, we define a morphism $f = (f_0, f_1, \ldots, f_l) : N \to M$ by setting $f_k = j$ and setting all other f_i to be the appropriate identity functions. Similarly define $g = (g_0, g_1, \ldots, g_l) : M \to N$ by setting $g_k = \pi$ and setting all other g_i to be the appropriate identity functions. Both are indeed morphisms, since the squares in the above diagram commute, and we have $g \circ f = 1_N$. Thus, f is a split monomorphism and g a split epimorphism.

We have thus added an inactive node to N, or viewed in reverse, we have removed an inactive node from M. By composing with a permutation, we may add or remove an inactive node at any position in the layer, and this process also yields a split mono and epi pair. By composing multiple such split monos or epis, we can see that adding or removing any number of inactive nodes at any layers results in a split mono and epi pair between the old and new networks.

This example is perhaps less trivial than it may look at first: dropout is a common technique that randomly chooses a subset of nodes to be inactive during each step of the training process. This technique is motivated by the idea that the remaining nodes form a network with simpler architecture that will be trained to perform the same task as the full network: the hope is that training many of these simpler networks simultaneously results in a more robust full network. The split mono/epi pairs described above are the algebraic description of a network with simpler architecture embedding into a larger network with inactive nodes.

VIII. General Monomorphisms and Epimorphisms

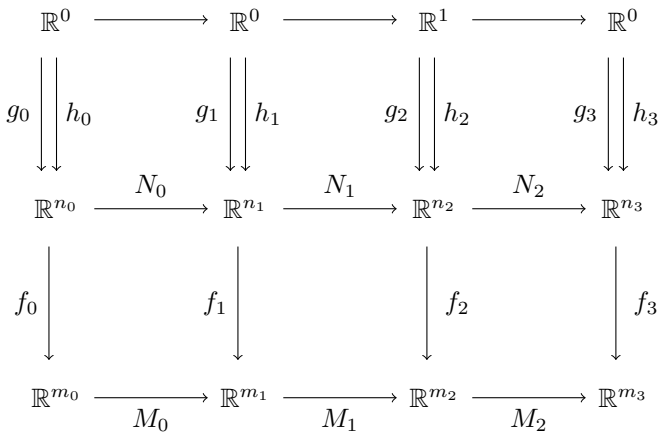
We now characterize monomorphisms and epimorphisms. In both categories, a morphism $(f_0, f_1, \ldots, f_l)$ is a monomorphism if and only if each f_i is injective, and is an epimorphism if and only if each f_i is surjective. While these results are not unexpected, the proofs of the "only if" parts of these statements are not entirely straightforward. They will make use of networks where one of the Euclidean spaces has dimension 1, and the rest have dimension 0. These will allow us to examine the behavior of a morphism at a single layer.

Theorem 2. *In both* ***AffineNet***$_l$ *and* ***ReluAffineNet***$_l$*, a morphism* $(f_0, f_1, \ldots, f_l) : N \to M$ *is a monomorphism if and only if each* f_i *is injective.*

Proof. If each f_i is injective, then for any morphisms $g, h : Z \to N$, if $f \circ g = f \circ h$, then $f_i \circ g_i = f_i \circ h_i$ for each i, which implies $g_i = h_i$ for each i. Thus $g = h$, so f is a monomorphism.

Conversely, suppose f is a monomorphism. We will first show f_0 is injective, by considering morphisms $g, h : T \to N$, where T is the terminal network. Suppose for a contradiction that $f_0(x_g) = f_0(x_h)$ for some $x_g \neq x_h$ in $\mathbb{R}^{n_0}$. Define g and h by setting $g_0(0) = x_g$ and $h_0(0) = x_h$. Then $f_0 \circ g_0(0) = f_0 \circ h_0(0)$, so for any i we have $M_{i-1} \circ \cdots \circ M_0 \circ f_0 \circ g_0(0) = M_{i-1} \circ \cdots \circ M_0 \circ f_0 \circ h_0(0)$. This implies $f_i \circ N_{i-1} \circ \cdots \circ N_0 \circ g_0(0) = f_i \circ N_{i-1} \circ \cdots \circ N_0 \circ h_0(0)$, which in turn implies $f_i \circ g_i(0) = f_i \circ h_i(0)$. But then $f \circ g = f \circ h$, while $g \neq h$, contradicting the assumption that f is a monomorphism. Therefore f_0 must be injective.

We now show f_k is injective for any $k > 0$. If $n_k = 0$ this is trivial, so suppose $n_k > 0$. We will consider morphisms $g, h : Z \to N$, where $z_k = 1$, $z_i = 0$ for all $i \neq k$, and each Z_i is constant with value zero. Note that this can be viewed as a ReLU network, so it is in both **AffineNet**$_l$ and **ReluAffineNet**$_l$. The following diagram shows an example with $k = 2$ for reference.



Let $x_g \neq x_h$ belong to $\mathbb{R}^{n_k}$. Define g and h as follows. Let $g_0(0) = h_0(0) = 0$, and for all $i > 0$ not equal to k, let $g_i(0) = h_i(0) = N_{i-1} \circ \cdots \circ N_0(0)$. Define $g_k(t) = tx_g + (1-t)N_{k-1} \circ \cdots \circ N_0(0)$ and $h_k(t) = tx_h + (1-t)N_{k-1} \circ \cdots \circ N_0(0)$. It can be checked that g and h are morphisms, and since $g \neq h$, we have $f \circ g \neq f \circ h$ because f is a monomorphism. In particular, this must mean $f_k \circ g_k \neq f_k \circ h_k$. We know $f_k \circ g_k$ and $f_k \circ h_k$ are of the form $f_k \circ g_k(t) = y_g + tv_g$ and $f_k \circ h_k(t) = y_h + tv_h$, and since $g_k(0) = h_k(0)$, we have $y_g = y_h$. Thus, $v_g \neq v_h$, so $f_k(x_g) = f_k \circ g_k(1) \neq f_k \circ h_k(1) = f_k(x_h)$. Therefore f_k is injective.

□

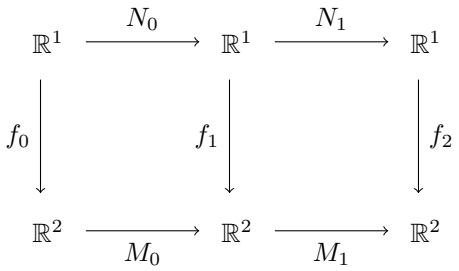
Theorem 3. *In both* ***AffineNet***$_l$ *and* ***ReluAffineNet***$_l$*, a morphism* $(f_0, f_1, \ldots, f_l) : N \to M$ *is an epimorphism if and only if each* f_i *is surjective.*

Proof. Once again, the "if" direction is straightforward, so for the "only if," suppose f is an epimorphism. We will first show that f_0 is surjective. Suppose otherwise, and consider morphisms $g, h : M \to Z$, where $z_0 = 1$, $z_i = 0$ for all $i \neq 0$, and each Z_i is constant with value zero. As noted before, Z is present in both categories. For all $i > 0$, g_i and h_i can only be constant with value zero. Since we have supposed f_0 is not surjective, the image $f_0(\mathbb{R}^{n_0})$ is an affine subspace strictly contained in $\mathbb{R}^{m_0}$. As such, we can choose distinct affine functions $g_0 \neq h_0$ that are equal on $f_0(\mathbb{R}^{n_0})$. It can be checked that g and h are then morphisms. Thus, $g_0 \circ f_0 = h_0 \circ f_0$, so we have $g \circ f = h \circ f$. But $g \neq h$, so this contradicts the fact that f is an epimorphism. Therefore f_0 is surjective.

For any $k > 0$, we use a similar argument: suppose for a contradiction that f_k is not surjective. Define morphisms $g, h : M \to Z$, where $z_k = 1$, $z_i = 0$ for all $i \neq k$, and each Z_i is constant with value zero. For each $i \neq k$, g_i and h_i must be constant with value zero. Again, $f_k(\mathbb{R}^{n_k})$ is a strict affine subspace of $\mathbb{R}^{m_k}$, so we may choose distinct affine functions $g_k \neq h_k$ such that $g_k(x) = h_k(x) = 0$ for all $x \in f_k(\mathbb{R}^{n_k})$. To show this choice gives g and h that are morphisms, we first note that since f is a morphism and f_0 is surjective, $M_{k-1} \circ \cdots \circ M_0(\mathbb{R}^{m_0}) \subseteq f_k(\mathbb{R}^{n_k})$. Thus, $g_k \circ M_{k-1} \circ \cdots \circ M_0$ and $h_k \circ M_{k-1} \circ \cdots \circ M_0$ are both constant with value zero, so g and h are in fact morphisms. By the construction of g_k and h_k, we have $g \circ f = h \circ f$, but $g \neq h$. Since f is an epimorphism, this is a contradiction, and we can conclude that each f_i is surjective. □

A monomorphism $f : N \to M$ can be interpreted as embedding one network in another. The morphism identifies a piece of the network M that behaves in the same way as N. Thus, if one has a such a monomorphism, an x that would normally be an input to N can be input to M after applying f_0. At each layer of M, the image $M_{i-1} \circ \cdots \circ M_0 \circ f_0(x)$ can be translated back to N since each f_i is injective. In particular, the output satisfies $M_{l-1} \circ \cdots \circ M_0 \circ f_0(x) = f_l \circ N_{l-1} \circ \cdots \circ N_0(x)$ and f_l is injective, so we may recover $N_{l-1} \circ \cdots \circ N_0(x)$. Thus, M can perform the same task as N, up to affine maps between the layers. There are likely ways to expand on this idea – some are suggested in the section on ideas for future work.

A simple example of a monomorphism is given below.



$$N_0(x) = \sigma(x-1)$$
$$N_1(x) = \sigma(2-x)$$

$$M_0(x) = \sigma\left(\begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix} x + \begin{pmatrix} -1 \\ -1 \end{pmatrix}\right)$$
$$M_1(x) = \sigma\left(\begin{pmatrix} 1 & -1 \\ -1 & -1 \end{pmatrix} x + \begin{pmatrix} -3 \\ 3 \end{pmatrix}\right)$$
$$f_0(x) = \begin{pmatrix} 3 \\ 2 \end{pmatrix} x + \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$
$$f_1(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$f_2(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} x$$

IX. Products

We now construct binary products in both categories and give an intuitive interpretation of a product of two networks.

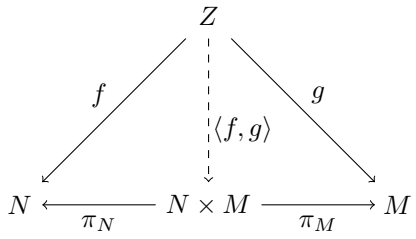
Theorem 4. *Both **AffineNet**$_l$ and **ReluAffineNet**$_l$ have all binary products.*

Proof. Given two networks N and M, we will show that their product $N \times M$ is given by

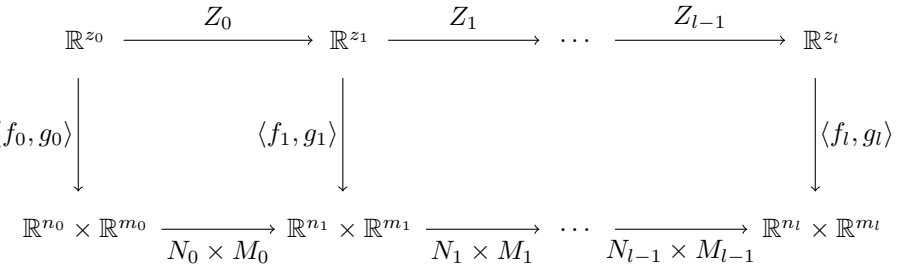
$$\mathbb{R}^{n_0} \times \mathbb{R}^{m_0} \xrightarrow{N_0 \times M_0} \mathbb{R}^{n_1} \times \mathbb{R}^{m_1} \xrightarrow{N_1 \times M_1} \cdots \xrightarrow{N_{l-1} \times M_{l-1}} \mathbb{R}^{n_l} \times \mathbb{R}^{m_l}.$$

The projection $\pi_N : N \times M \to N$ is then $\pi_N = (\pi_{N,0}, \pi_{N,1}, \ldots, \pi_{N,l})$, where each $\pi_{N,i} : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$ is the projection onto $\mathbb{R}^{n_i}$. The projection $\pi_M : N \times M \to M$ is defined similarly. If we are working in **AffineNet**$_l$, it can be checked that the product maps $N_i \times M_i$ are affine functions followed by some activation function. If we are working in **ReluAffineNet**$_l$ these activation functions are in fact ReLU. It can also be checked that the projections are morphisms.

We now verify that the product and projections satisfy the universal mapping property of products. Let Z be a network and let f and g be morphisms as in the diagram.



We construct the morphism $\langle f, g \rangle$ by setting $\langle f, g \rangle_i = \langle f_i, g_i \rangle$, as in the following diagram.



Each $\langle f_i, g_i \rangle$ is an affine map since f_i and g_i are, and it

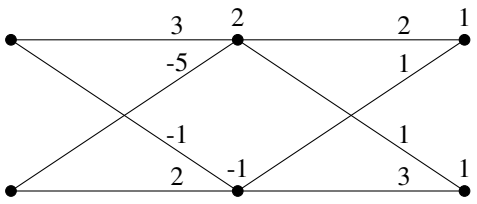
can be verified that all rectangles starting at $\langle f_0, g_0 \rangle$ commute because f and g are morphisms. Therefore $\langle f, g \rangle$ is an affine morphism. Each $\langle f_i, g_i \rangle$ is the unique function such that $\pi_{N,i} \circ \langle f_i, g_i \rangle = f_i$ and $\pi_{M,i} \circ \langle f_i, g_i \rangle = g_i$, so we can conclude that $\langle f, g \rangle$ is the unique morphism such that $\pi_N \circ \langle f, g \rangle = f$ and $\pi_M \circ \langle f, g \rangle = g$, as required.

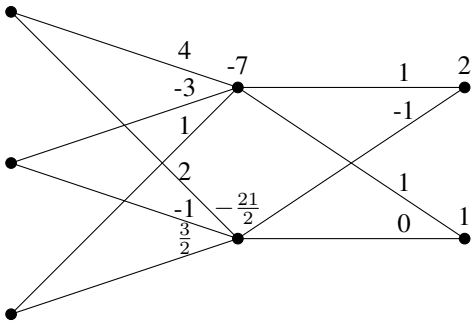
□

This construction of the product of two neural networks has a natural interpretation: it is the network that simply places the two networks side by side, without letting them interact, and performs their functions on pairs of inputs.

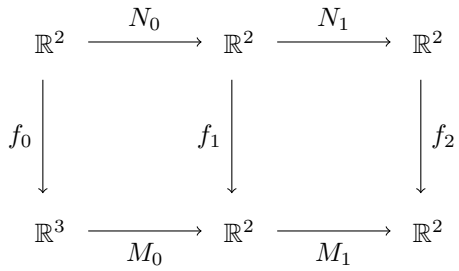
X. A Final Example

We now present one last example that does not include the previous specific types of objects or morphisms. The morphism given here demonstrates that nontrivial morphisms can exist between networks that may at first look unrelated. The networks are shown below.





A morphism from the first network to the second is as follows.



$$N_0(x) = \sigma\left(\begin{pmatrix} 3 & -5 \\ -1 & 2 \end{pmatrix} x + \begin{pmatrix} 2 \\ -1 \end{pmatrix} \right)$$

$$N_1(x) = \sigma\left(\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} x + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right)$$

$$M_0(x) = \sigma\left(\begin{pmatrix} 4 & -3 & 1 \\ 2 & -1 & \frac{3}{2} \end{pmatrix} x + \begin{pmatrix} -7 \\ -\frac{21}{2} \end{pmatrix} \right)$$

$$M_1(x) = \sigma\left(\begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} x + \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right)$$

$$f_0(x) = \begin{pmatrix} -1 & \frac{5}{3} \\ -2 & \frac{10}{3} \\ 1 & -\frac{5}{3} \end{pmatrix} x + \begin{pmatrix} \frac{19}{3} \\ \frac{20}{3} \\ \frac{11}{3} \end{pmatrix}$$

$$f_1(x) = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 0 \end{pmatrix} x$$

$$f_2(x) = \begin{pmatrix} \frac{3}{10} & -\frac{1}{10} \\ \frac{3}{5} & -\frac{1}{5} \end{pmatrix} x + \begin{pmatrix} \frac{9}{5} \\ \frac{3}{5} \end{pmatrix}$$

Note that the image of each f_i is a 1-dimensional affine subspace, so the morphism is not a monomorphism, nor an epimorphism. Even in this relatively small example, verifying that this is a morphism by hand is a long process, so for future work, morphisms between large networks will likely need to be handled by computers.

XI. Conclusion and Ideas for Future Work

The work presented here can hopefully serve as a starting point for further deeper analysis of neural networks. We have defined particular categories of neural networks and shown that they have certain types of objects and morphisms commonly considered in category theory. Moreover, these category theoretic constructions generally have useful interpretations and convey information about the networks involved.

Some ideas for future works are listed below.

1) The categories defined here only contain networks of one length. It may be possible to extend these definitions to allow networks of different lengths.
2) Finding nontrivial examples of morphisms between realistic neural networks will be necessary to make this work relevant.
3) In some situations it may be mathematically convenient to allow networks N with the empty set in place of $\mathbb{R}^{n_i}$, for one or more i. This makes sense in the context of affine functions, as the empty set can be considered as an affine space. Whether this is truly useful can be the subject of future work; for now, we will suggest the notation $\mathbf{AffineNet}_l^{\varnothing}$ for the category defined in the same way as $\mathbf{AffineNet}_l$, except allowing $\varnothing$ in place of any $\mathbb{R}^{n_i}$ in a network and allowing a function N_i to be the empty function if its domain is $\varnothing$. An f_i of a morphism f is also allowed to be the empty function if its domain is empty. We can also form $\mathbf{ReluAffineNet}_l^{\varnothing}$, defined in the same way as $\mathbf{ReluAffineNet}_l$, except allowing the same substitutions of empty sets and empty functions.
4) Other limits and colimits appear to be difficult to handle in these categories. The construction of the product might suggest that a layer by layer construction could work for, say, equalizers. In $\mathbf{AffineNet}_l$ and $\mathbf{ReluAffineNet}_l$ this is not the case: an equalizer of two affine functions can be empty. This construction also fails for other specific morphisms in $\mathbf{ReluAffineNet}_l$, in which the resulting network would be required to have non-ReLU activations. Coequalizers appear to have similar problems in $\mathbf{ReluAffineNet}_l$. One could examine whether all equalizers exist in $\mathbf{ReluAffineNet}_l^{\varnothing}$, as this fixes the problem of empty equalizers of affine functions. Perhaps a more interesting question is which equalizers do exist in these categories and whether any are of particular interest.
5) Monomorphisms may give a language to talk about "subnetworks." A category theoretic approach to handling subobjects in a specific category is to simply define subobjects as monomorphisms. Note that this will generally differ from the intuitive idea of forming a "subnetwork" by considering only a subset of nodes and their connections. Defining a subnetwork as a monomorphism would allow us to consider smaller networks that respect the behavior of the original network. One can then ask whether there is a way to define the "image" of a morphism as a subnetwork.
6) It is important to keep in mind that realistic neural networks will not usually involve weights that are integers or simple rational numbers as are used in examples here. In addition, changing a weight by a very small amount could result in a new network that is not isomorphic to the original, while computationally their performance is indistinguishable. It may be beneficial to have some notion of an approximate morphism in order to handle realistic networks that do not fit into the exact mathematical definitions.
7) While examining categories with affine morphisms seems to be a useful place to start, there may be other classes of morphisms that could prove useful. Very broadly, other subcategories of the first general definition could be considered.
8) Among all ReLU networks of the form $\mathbb{R} \xrightarrow{\sigma \circ A_0} \mathbb{R} \xrightarrow{\sigma \circ A_1} \mathbb{R}$, it appears that there are only a finite number of isomorphism classes, determined by the nature of the affine functions A_0 and A_1. This also seems plausible for any given shape of ReLU network. It may be interesting to determine whether this is true, if there is a reasonable way to count isomorphism classes for a given shape, and if ReLU networks could systematically be identified by isomorphism class. This would open up the possibility of studying ReLU networks based on isomorphism classes, including examining the relationship between isomorphism classes of various shapes of networks.
9) In many realistic cases, the type of data input to a network N is limited to a bounded subset of $\mathbb{R}^{n_0}$. With image data, for instance, each pixel is input to a network as a number in some bounded interval, so an image is contained in a box in high dimensional space. It would be possible to consider another category in which a network's layer functions can be defined on subsets of Euclidean space.
10) As a "stretch goal," we would like to work towards a covering family, and perhaps to see if it permits the Grothendieck construction of K_0.
11) The nature of isomorphic neural networks in this context is not well-understood. One could train a (relatively small) neural network and create isomorphic neural networks from it – these could then be used to study how isomorphic neural networks respond to the same input and to further training.
12) By putting neural networks in a categorical framework, other work towards making categorical structures *temporal* can be leveraged. Thus this effort might facilitate a study into e.g. temporal graph neural networks.

REFERENCES

[1] B. Fong, D. Spivak, and R. Tuyéras, "Backprop as functor: A compositional perspective on supervised learning," pp. 1–13, 2019.
[2] M. J. Healy and T. Caudell, "Neural networks, knowledge and cognition: A mathematical semantic model based upon category theory," 2004.
[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.
[4] S. Awodey, *Category Theory*. New York: Oxford University Press Inc., 2012.
[5] S. Mac Lane, *Categories for the Working Mathematician*. New York: Springer-Verlag New York, Inc., 1998.
[6] C. C. Aggarwal, *Neural networks and deep learning: a textbook*. Springer, 2019.