Abstract

This note is to summarize the paper Geometric Deep Learning[1].

1 Introduction

DNN models are specified to deal with data that has very high dimensionality e.g. a 4K image has3840×2160 pixels, and each pixel has 3 channels, which gives$3840 \times 2160 \times 3$ number of independent variables. For a generic high dimensional function, the data needed to specify this function increases exponentially with the dimension. This fact is called curse of dimensionality because it simply forbids learning very high dimensional generic functions. A DNN model based on universal approximation theorem is doomed with curse of dimensionality as well. However, in real life applications, the function to be learned is not generic. Thus we must regulate the DNN model i.e. introduce non-genericity to the model. This non-genericity comes from human prior knowledge of the function to be fitted by a DNN model. One example is the symmetry of the function. If we know in prior that the function is invariant under certain symmetry group transformation of the input variables, this symmetry requirement will constrains the space of parameters of a DNN model.

2 Category view of DNN model

Definition 1. (Category). A category,$\mathcal{C}$, consists of a collection of objects and a collection of morphisms between paris of objects, such that:

- For each object$A \in \mathcal{C}$, there is a unique identity morphism$\mathrm{id}_A : A \to A$.
- For any two morphisms$f : A \to B$ and$g : B \to C$, there must exist a unique morphism which is their composition$g \circ f : A \to C$.

subject to the following conditions

- For any morphism$f : A \to B$, it holds that$\mathrm{id}_B \circ f = f \circ \mathrm{id}_A = f$.
- For any three composable morphims$f : A \to B$,$g : B \to C$,$h : C \to D$, composition is associative, i.e.,$h \circ (g \circ f) = (h \circ g) \circ f$.

Taking the MLP as an example, we may view it as a category with object being linear spaces and morphism being non-linear transformation between the linear spaces. Two linear spaces are specially choosen as the input and output spaces. Other linear spaces are the output space of hidden layers. The morphism is the combination of linear transformation, bias addition, and activation function. A special morphism from input linear space to output linear space exists by definition from the successive composition of morphism beween input-hidden, hidden-hidden, hidden-output linear spaces. This particular morphism called“inference morphism” is the one choosen to map input to output despite other choices. To represent such a"MLP category", we usually do not need to write down every possible morphism. We can list all linear spaces and a subset of all morphisms called"morphism generator set". The rest of the morphism can be achieved from composition.

Due to the internal structure of linear space, a pair of linear spaces can be mapped to another linear space by direct sum labelled by$\oplus$ or tensor product labelled by$\otimes$. These two extra structure will complicate the category further which will be deferred into later for discussion.

We may describe a DNN model as a category denoted by**VectNonLinear**. The objects in**VectNonLinear** are vector spaces and the morphisms in**VectNonLinear** are general non-linear transformations of vectors. Due to the internal structure of linear space, a pair of linear spaces can be mapped to another linear space by direct sum labelled by$\oplus$ or tensor product labelled by$\otimes$. These two extra structure will complicate the category further which will be deferred into later for discussion. A key difference between**VectNonLinear** and**Vect** (the category of all vetor spaces) is that the morphism is general non-linear transformations. Two vector spaces are specially identified as the input data vector space$I$ and the output data vector space$O$. The rest of the vector spaces are output and input vector space of hidden layers.

Each morphism in**VectNonLinear** are non-linear transformations from one vector space to another defined by how it map each vector in the domain vector space to the vector in codomain vector space. The identity morphism for each vector space is simply the identity function that maps each vector into itself. If we view the**VectDNN** as a directed graph with node being vector space and directed edge being morphism, by definition, if we can start from one node and travelling along the directed edge to another node (possibly passing several other nodes along the path), there must exist an single directed edge that connects the origin node and the destination node. This might shed some light on the functional programming of a**VectDNN**. Usually only part of the functions between vector spaces are provided manually in code, but the definition of category requires functions which are composition of existing provided functions exist uniquely. This hints that new function code might need to be generated based on existing ones.

Definition 2. (Functor). Let$\mathcal{C}$ and$\mathcal{D}$ be two categories. Then,$F : \mathcal{C} \to \mathcal{D}$ is a functor between them, if it maps each object and morphism of$\mathcal{C}$ to a corresponding one in$\mathcal{D}$, and the following two conditions hold:

- For any object$A \in \mathcal{C}$,$F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$.
- For any composable morphismsf, g in$\mathcal{C}$,$F(g \circ f) = F(g) \circ F(f)$.

A functor is called faithful if the functor map is injective and full if surjective. A bijective functor is called fully faithful.

Definition 3. (Concrete Category). A concrete category is a pair$(\mathcal{C}, U)$ satisfying

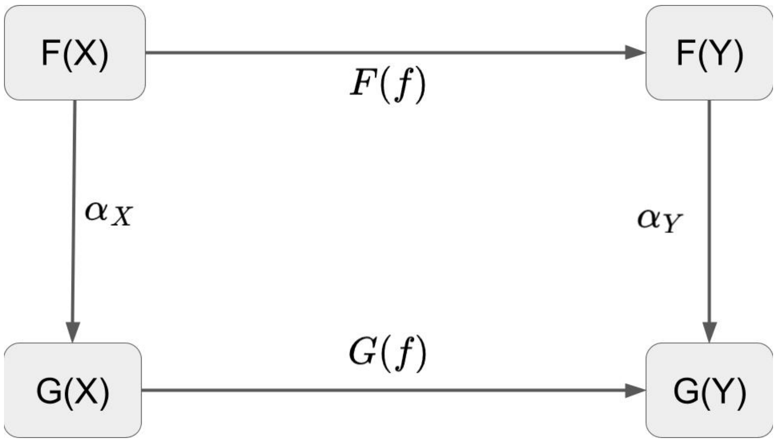


Figure 1: Natural Transformation

- $\mathcal{C}$ is a category
- $U : \mathcal{C} \to \mathbf{Set}$(the category of sets and functions) is a faithful functor

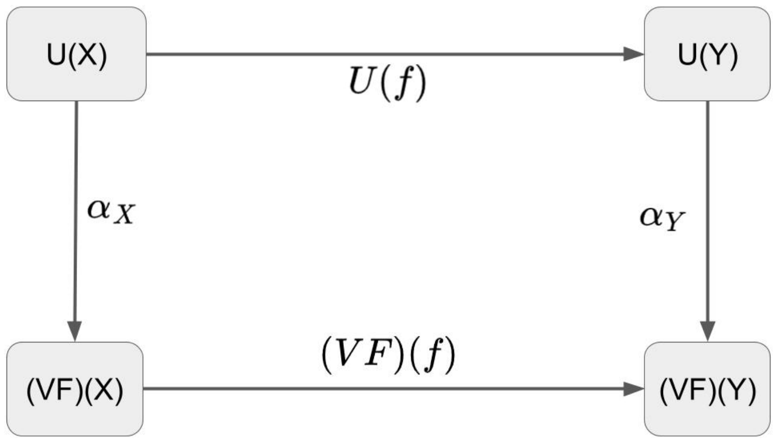
U grasps the set strucuture of the category$\mathcal{C}$, which might be part of the total structural information of$\mathcal{C}$, thus$U$ is also called forgetful function. For any morphism$f : X \to Y$ in$\mathcal{C}$,$U(f)$ is a function with domain$U(X)$ and codomain$U(Y)$. In the following discussion when$\mathcal{C}$ is the**VectDNN**, we chose$U(X) = \{x|x \in X\}$ for any linear space$X$ in**VectDNN** which forgets the scalar multiplication and vector addition of a vector space structure. Since in**VectDNN**, the original morphism$f : X \to Y$ has information about how each vector in$X$ is mapped to which vector in$Y$, then$U(f)$ is the corresponding function in**Set** that keeps this information and nothing else. This choice of$U$ is obviously injective or faithful.

The following discussion shows that each equivariant condition of a DNN can be described as the**VectDNN** admits a corresponding concrete endofunctor that leaves the non-linear transformation between vector spaces invariant. Endofunctors invovles understanding relationship between two functors.

Definition 4. (Natural Transformation). Let$F : \mathcal{C} \to \mathcal{D}$ and$G : \mathcal{C} \to \mathcal{D}$ be two functors between categories$\mathcal{C}$ and$\mathcal{D}$. A natural transformationα from$F$ to$G$ assigns to each object$X$ in$\mathcal{C}$ a morphism$\alpha_X : F(X) \to G(X)$ in$\mathcal{D}$ such that for every morphism$f : X \to Y$ in$\mathcal{C}$,$\alpha_Y \circ F(f) = G(f) \circ \alpha_X$.

Given a natural transformation and a functor$F$ as defined above, functor$G$ cannot be uniquely determined due to$G(f)$ cannot be solved in general from the requirement$\alpha_Y \circ F(f) = G(f) \circ \alpha_X$. Thus natural transformation cannot be simply understood as a map from functor to functor.

Definition 5. (Concrete Functor). Let$(\mathcal{C}, U)$ and$(\mathcal{D}, V)$ be two concrete categories. A concrete functor is a functor$F : \mathcal{C} \to \mathcal{D}$ such that there exists a natural transformation from$U$ to$V \circ F$.



$$\begin{tikzcd} U(X) \arrow[r, "U(f)"'] \arrow[d, "\alpha_X"] & U(Y) \arrow[d, "\alpha_Y"'] \\ (VF)(X) \arrow[r, "(VF)(f)"] & (VF)(Y) \end{tikzcd}$$

Figure 2: Concrete Functor

All element appears in the natural transformation diagram (2) is either set or function between sets. Thus the commutation of the two paths are familiar equation of functions we all learnt at calculas:

$$\alpha_Y((U(f))(x)) = (V \circ F)(f)(\alpha_X(x)), \tag{1}$$

for arbitrary indepedent variable$x$ in the domain$U(X)$.

With the above setup, an equivariant DNN model is described as a concrete**VectDNN** that admits a concrete endofunctor that maps any morphism in**VectDNN** to itself. As discussed**VectDNN** is a concrete category. IfF is a endofunctor, then$V$ on the right hand side of Eq.(1) should be replaced by$U$. Further more, if$F$ leaves any morphism invariant$(U \circ F)(f) = U(f)$. Then Eq.(1) becomes:

$$\alpha_Y((U(f))(x)) = U(f)(\alpha_X(x)), \tag{2}$$

although$F$ is explicitly evident in the above expression,α_X,α_Y is closely associated with$F$. Recall that$f : X \to Y$ is interpreted as the non-linear transformation of a DNN layer, Eq.(2) shows thatα_Y is the equivariance transformation of the ouput of the layer when the input is transformed by the corresponding equivariance transformationα_X. Thus we reach a rigorous mathematical defintion of equivariance within category theory.

3 Complete definition

Last section gives a simplified version of using category theory to desribe the DNN model by ignoring the direct sum and tensor product of linear spaces. In this section, we add those two additional structure to the definition.

rig category,

Definition 6. (Monoidal Category) A monoidal category is a category$\mathcal{C}$ equipped with a monoidal structure. A monoidal structure consists

$$\begin{tikzcd}
U(X) \arrow[r, "U(f)"'] \arrow[d, "\alpha_X"] & U(Y) \arrow[d, "\alpha_Y"'] \\
(UF)(X) \arrow[r, "(UF)(f)"] & (UF)(Y)
\end{tikzcd}$$

Figure 3: Concrete EndoFunctor

4 Constructing Equivariant VectDNN

Previous Section starts from formal definitions and reveals the categorical language needed to describe a equivariant DNN model. In this section we address the more down to earth pratical needs: how to construct such a DNN model given the equivariance constraint. In practice, we are not given the**VectDNN** and constructs concrete endofunctor that leaves its morphism invariant, on the contrary, we are given the endofunctor$F$ and try to construct such a concrete category**VectDNN**. In terms of Fig.(3), the goal if to find$f$ that the makes the diagram commutes with the rest of the elements in the diagram fixed. Say we have chosen an arbitrary$f$, to make the diagram commute,$(UF)(f)$ is automatically determined.

5 Group Equivariance

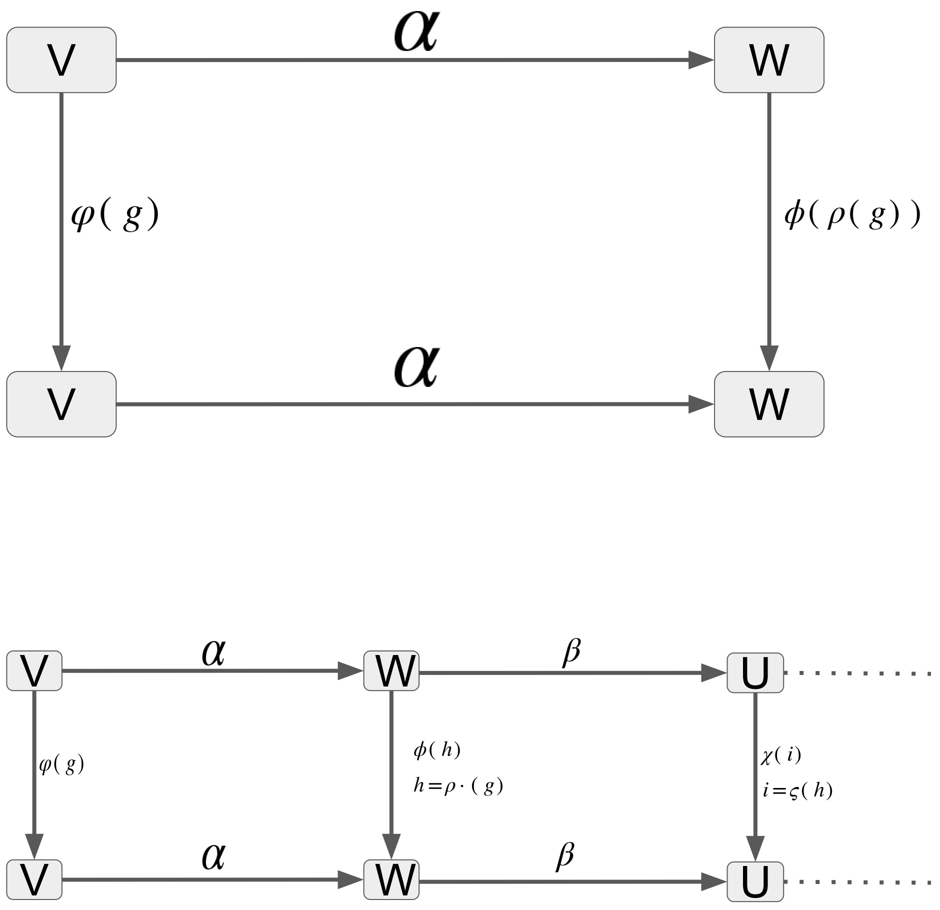
Without losing generality, the input$X$ and output$Y$ of a DNN model can be viewed two vectors:

$$X \in R^n, \quad Y \in R^m. \tag{3}$$

Althought there are different types prior knowledge can be imbued into the model, here we focus on symmetry prior i.e. the input and output are equivariant under a homomorphism mapping between group representations illustrated by figure (4).

Definition 7. If G and H are two groups related by a homomorphism$\rho : G \to H$ and$\varphi : G \to GL(V)$ and$\psi : H \to GL(W)$ are representations, then an intertwining map or equivariant map$\alpha : V \to W$ satisfies:

$$\alpha \circ \varphi(g) = \psi(\rho(g)) \circ \alpha \tag{4}$$



$$\begin{tikzcd}
V \arrow[r, "\alpha"] \arrow[d, "\varphi(g)"] & W \arrow[d, "\phi(\rho(g))"] \\
V \arrow[r, "\alpha"] & W
\end{tikzcd}$$

$$\begin{tikzcd}
V \arrow[r, "\alpha"] \arrow[d, "\varphi(g)"] & W \arrow[r, "\beta"] \arrow[d, "\phi(h)" , "h=\rho\cdot(g)"'] & U \arrow[r, dotted, no head] \arrow[d, "\chi(i)", "i=\varsigma(h)"'] & {} \\
V \arrow[r, "\alpha"] & W \arrow[r, "\beta"] & U \arrow[r, dotted, no head] & {}
\end{tikzcd}$$

Figure 4: Equivariant diagrams of two homomorphic groups$G$ and$H$ with their representations.

If the input vector$X$ under goes some symmetry tranformation, we know in advance that the output$Y$ should also be transformed by the corresponding symmetry transformation.Y being invariant is a special case where all symmetry transformation of$X$ is mapped to the identity transformaiton of$Y$. This means that the DNN model is a equivariant map as defined above. This puts constraint on the possible structure of the DNN model which both decrease the degrees of freedom of the model and increase the generalibility of the model. However, for complex models, it is not easy to translate the equivariance property into detailed contraints on the model. For DNN due to its layered structure, a practical approach is to require equivariance of each layer and design the layers such that it"reduces" to the desired equivariance for the final output. Here the word"reduces" is used asusming in general the output adimits a group of lower order. However the discussion applies equally when the output group has the same order as the input group.

Given group$G$, according to the fundamental homomorphism theorem,$H \cong G/Ker(\rho)$, i.e.,H is isomorphic to the quotient group of$G$ with the subnormal group being the kernel of the homomorphism.

In the following we use the group symbol$G$ and$H$ with its representation interchangably only specify their difference when necessary. We use small letter for vector and capital letter for matrix. Then the equivariance requirement for each layer reads:

$$H\alpha(x) = \alpha(Gx), \quad \forall H = \rho(G), \quad x \in V. \tag{5}$$

There are two difficulties to pratically calculate the above relation. First, in general the order of group$G$ and$H$ can be very lalrge. Second, the mappingα in general is non-linear. As pointed out in ([2]), it is sufficient to use the generators of group$G$. Since$H$ is homomorphic to$G$, a complete generator of$G$ is also a complete generator of$H$, so is the representations of the generators. Then instead of studying every pair of$H = \rho(G)$, we only need to study the generators which is far less than the order of a group.

Suppose we have two equivariance mapα_1 andα_2, we can study their relationships. Supposeα_1 is related toα_2 by a invertible map$T$

$$\alpha_1 = T(\alpha_2), \tag{6}$$

using relation (5), we have

$$\begin{aligned} \alpha_1(Gx) &= T(\alpha_2(Gx)) \\ HT(\alpha_2(x)) &= T(H\alpha_2(x)), \end{aligned}$$

thus$T$ is a equivariance map from vector space$W$ to itself with the group$H$ trivial endomorphism. Thus it can be seen that adding a equivariant layer from vector space W to itself and group H trivial endomorphism is redundant if the equivariance layer is to be learnt. It simply transform the unknown equivariance mapα to another unknown possible solution. Nonetheless, it can be used to generate new equivariance maps given a solution.

IfT is a linear transformation, then in terms of its matrix form we simply have

$$HT = TH \tag{7}$$

where both$H$ and$T$ are matrix. Thus given a genericα, all$T$ matricies satifying the above relation gives rise to a class of equivariance map.

A common emerging problem is that the function form of equivariance map may not align with the popular structure of a single layer perceptron.

6 Different groups

6.1 Permutation group

Any permutation groupS_n can be generated by two elements:$(1,2)$ and$n$-cycle$(1,2,3,\ldots,n)$. The$n$-cycle can move any two adjacent elements to the first and second position to be swapped by$(1,2)$, then sent back to their original positions by continue$n$-cycle. Thus any adjacent2-cycle can be generated. Then using adjacent2-cycle, other2-cycle can be generated by repeatingly swap from element$i$ towards another element$j$, swapi,j, then swap$j$ to$i$th position. Then using the fact the any element of ofS_n can be decomposed into2-cycles, it is proved that the two elements$(1,2)$ and$n$-cycle$(1,2,3,\ldots,n)$ are minimal complete generators. For the general equivariance map:

$$Y^i = \alpha^i(X^1, X^2, \ldots, X^n), \tag{8}$$

we only need to specify how the component of$Y$ changes when the two elements$(1,2)$ and$n$-cycle$(1,2,3,\ldots,n)$ are applied to$X$. However, there is no general way to find the image of$(1,2)$ and$(1,2,3,\ldots,n)$. Suppose the image of$(1,2)$ and$(1,2,3,\ldots,n)$ is choosen by analyzing the quotient group of$G$, with the corresponding image$\sigma_{(1,2)}$ and$\sigma_{(1,2,\ldots,n)}$, we thus have

$$\begin{aligned}\sigma_{(1,2)}(Y^i) &= \alpha^i(X^2, X^1, \ldots, X^n),\\ \sigma_{(1,2,\ldots,n)}(Y^i) &= \alpha^i(X^n, X^1, \ldots, X^{n-1}).\end{aligned} \tag{9}$$

Consider a generic multiceptron with no hidden layer

$$Y^i = act(W^i_j X^j), \tag{10}$$

6.2 Poincare group

7 A novel idea

If using the category graph, we can view a DNN model as a graphical representation of a total distribution conditioned on the input space. We may generalize the delta function distribution on each edge to e.g. a gaussian distribution. Then we need to integrate to get the final conditional distribution of output given input. A advantage is that now we can invert the total conditioanl distribution to get a generative model. We find a way to convert any existing model into a generative model.

References

[1] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021.

[2] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *CoRR*, abs/2104.09459, 2021.