Neural Concept Shape (NCS) Q&A

General

1) How does neural concept shape work?

The core technology of Neural Concept shape are 3D convolutional networks that learn to predict the output of physical simulations or experiments based on the input shape's geometrical properties. They can be used as complements to the traditional numerical simulation and experimental methods so as to alleviate the need for actual simulations or experiments along the conception pipeline. This dramatically accelerates and reduces the cost of the design process.

This proprietary deep learning technology is also fully integrated with state-of-the-art shape optimization algorithms. Therefore, when relevant, the conception process can even be completely automated so as to find the optimal design(s) given a number of predefined objectives.

2) What types of physics can I use Neural Concept Shape for?

Our technology can be applied naturally on most of the data produced by engineers. In particular, we have had success with the following types of physics:

- External Aerodynamics. Both compressible and incompressible.
- Hydro-Dynamics. Including cavitation.
- Electro-magnetics.
- Optics.
- Thermal transfer.
- Additive manufacturing process simulation.
- Aerodynamics with sensors for outdoor spaces.

Generally speaking, it is now clear that NCS can perform well to reproduce any type of simulations based on Finite Element Analysis. It has also proven to be very efficient in learning from the correlations between experimental data and input shapes, in a fully data-driven way that is completely agnostic to the equations of the phenomena at stake.

The fully data-driven approach of NCS is also particularly suited for highly complex systems, such as multi-physics simulations, since it encapsulates most of the complexity within the learning process and offers engineers a simplified interface to interact with the physical system.

3) What dataset size do I need to have to train an accurate network?

This is a crucial question and the answer can vary a lot from one dataset to the other. But we can give some general answer from our experience. Generally, learning output field or sensor values is much more efficient than learning output scalar values — since we implicitly have multiple data points per geometry in the case of fields—. For fields, we can have usable results starting from 50 shapes. For scalars, we need around 300 samples minimum.

The performance keeps increasing dramatically as we add more data up to a few thousands examples and then starts converging.

When accurate simulation or sensor data is difficult to acquire and not many samples are available, it is good to use a pre-trained Network (see Q.9) to initialize the training — either trained from lower fidelity simulations or from NC's own Network base —. It can help reducing the required dataset size by a factor of 10.

4) How can I feed data to NCS?

As depicted in Fig 1., the CNN can use three types of input. You can choose to use any of these three or any combination of them depending on the case:

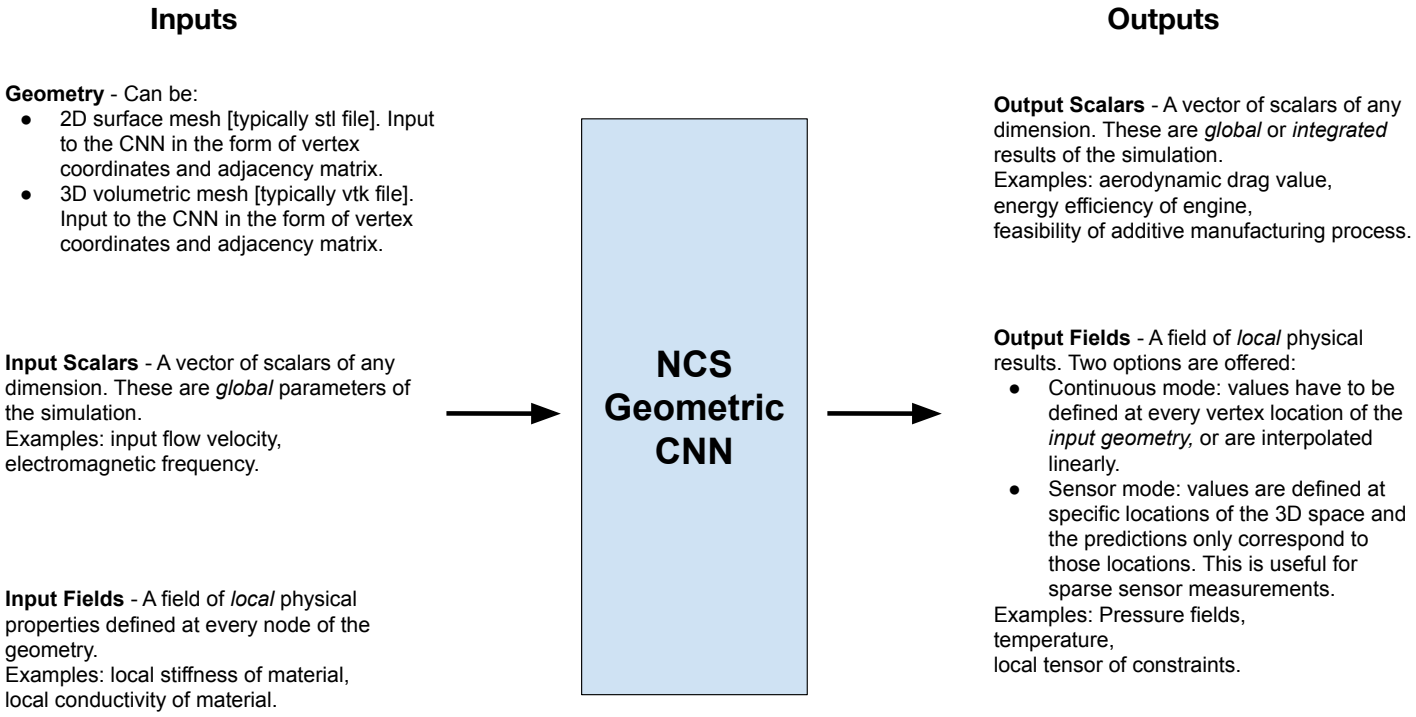

Fig 1: NCS Geometric CNN Input/Output

Remarks:
- the deep learning technology is based on a fully supervised learning process. In the training dataset, you will need to provide both the inputs and the outputs. Once the network is trained, you can use it on new data where you only have the inputs.
- In practice "fields" do not need to be defined at every point of the mesh. They can be defined sparsely on the geometry. Even when trained on sparse data, the network will still output dense predictions. You can even mix sparse and dense data in the same dataset. This feature is especially useful when working with sensor data.

Example: 3D airfoil simulation
To train a surrogate model to perform CFD simulations for a 3D airfoil you could use the following data structure:
Inputs:
- geometry: 2D surface mesh representing the mesh
- Input scalars: Reynolds number

Outputs:
- output fields: Cp
- Output scalars: lift-to-drag

Remark: if your simulations are all run with the same Reynolds number, you can avoid to provide it as an input scalar, and it will then be learned implicitly as a constant by the network.

5) What mesh sizes can NCS handle?

This is also one of the core elements we have worked on and we are now able to handle meshes with more than 2 million surface vertices on a single GPU, without a strong impact on the training or inference time.

The convolutions that we use can be considered a “generalization” of standard ones for images and hence are not necessarily more difficult to train.

6) Can NCS handle different mesh topologies?

Yes. This is the main strength of NCS, as we can handle completely arbitrary topologies within the same dataset. There can be a different number of connected components, different number of vertices, different topologies and types of elements.

Concretely, a single dataset can contain shapes *with* and *without* holes, and shapes that have been generated using completely different parametrizations. Actually, no parametrization is needed, if the shapes have been created manually for instance.

How we do that is the “magic sauce” of Neural Concept. We are using a selection public and private of state-of-the-art methods in Geometric Deep-Learning, for which we have very efficient custom CUDA implementations. Our expert knowledge and specific in-house techniques allow us to provide you with the most versatile and performant deep-learning-enhanced engineering platform currently available.

Deep learning

7) What special features does NCS offer for the training of Geometric Convolutional Neural Networks?

On top to the core library that offers custom proprietary Deep-Learning architectures, NCS offers a set of utilities to train Geometric CNNs efficiently. NCS was fully thought with the goal of making the training of your CNN first-time right.

Input and output normalization:
NCS automatically normalizes the input and outputs in order to make sure that learning capacity is allocated to all outputs equally. For example: Error of 30% for flow field at one single node is not as critical as 30% error for drag and this normalisation takes care of this.

Training utilities:
Thanks to NCS, the engineer does not need to worry about saving and restoring Neural Network trainings as it is automatically managed within the framework using a database.

High-Level definition of models:
With NCS, the engineer can use pre-defined Neural Network architectures that were optimized by our researchers, but he can also progressively dive into the definition of models through our configuration file formats.

Data formats:
In NCS, the engineer can very easily switch between data formats, use sensor or simulation types of data, modify and reuse a bank of pre-trained architectures that are readily available.

8) How long does it take to train a new neural network in NCS?

The training time depends on the level on complexity that is configured for the network, as well as the size of the data used for training. Therefore, it can vary from a use case to another.

A typical training time is 24h on a modern GPU (e.g Nvidia P100). Once trained, this network can then be used as a surrogate model to run simulations on new shapes in a matter of milliseconds.

9) Do I need to re-train a new network from scratch for every new simulation or experiment setup?

No. Transfer learning is widely supported by our framework. Continuing the example given in Q.3), let us assume you have trained a network N1 with a constant Reynolds number R1 and that you neglected to give the Reynolds as input scalar — because it was constant —, and now you have a new dataset with different Reynolds R2. Then, you can use use N1 to initialize the training of a new network N2 on the new dataset, which is going to be much faster and data efficient than starting from scratch.

10) I successfully trained a model on my dataset. What can I now do with it?

Once you have a trained model, you can use it to freely explore the shape space and obtain new simulation results in real-time without the need for any underlying parametrization. The only limit to your exploration is that the new shapes you use are within the validity domain of your model. For instance, in a Formula-1 case, if a network has only be trained with a front wing with 1 or 2 foils, it will not generalize to 5 foils at test time. Luckily, our Neural Networks are also able to give uncertainty estimations about the predictions they output. Therefore, you will know whenever you start getting out of the validity domain.

As mentioned in other answers, our deep-learning framework is also fully integrated with our optimization framework. Therefore, you also use trained models to automatically run shape optimization pipelines.

Optimization

11) What is special about Neural Concept's optimization method and how does it compare to other Bayesian optimization methods?

Our latest optimization library combines both 0-order optimization using advanced evolutionary algorithms (Genetic, CMA-ES), and first-order one using the availability of gradients end-to-end through the CNN during optimization.
It depends on the problems, but, yes, using gradients for optimization makes a big difference as soon as the parameters space is continuous and has dimension above 6 or 7.
Therefore, for simple and standard optimization problems with very small parameters space (below 6-7) and small number of simulations (below 10), and if there is no pre-training of the network available, Neural Concept will not bring real benefits compared to other commercially available Bayesian optimization methods. As soon as those numbers increase or that we can used a pre-trained Neural Network, Neural Concept will bring an exponentially growing benefit.

12) How different can the shapes obtained with optimization be compared to the initial training ones?

During the optimization process, the shapes that are explored will naturally drift away from the initial training dataset. Therefore, the accuracy of predictions for these shapes will drop. Hence, it is very important to be able to resample the simulator and feed the results back to training the network. Our recent results have shown that, by using such a procedure for iteratively retraining the network and optimizing, we can progressively, and quite quickly, explore new designs that are completely away from the initial sampling.

Then, yes, we can expect the optimization to converge toward significantly different designs that human minds wouldn't have thought of.