# Deep Neural Networks Are Effective At Learning High-Dimensional Hilbert-Valued Functions From Limited Data

Sebastian Moraga [1]
Ben Adock [1] Simone Brugiapaglia [2], Nick Dexter [1]
**smoragas@sfu.ca**
**sites.google.com/view/sebanthalas**

[1]Simon Fraser University. Canada
[2]Concordia University. Canada

July 20, 2021

# Outline

# Motivation

## Multivariate function recovery

Approximate $f : \mathcal{U} \to \mathcal{V}$, a Hilbert-valued function, from its evaluations at $m \in \mathbb{N}$ sample points $\boldsymbol{y}_1, \dots, \boldsymbol{y}_m \in \mathcal{U}$:

$$d_i = f(\boldsymbol{y}_i) + n_i \in \mathcal{V}_h, \qquad i = 1, \dots, m.$$

Input $\boldsymbol{y} \in \mathbb{R}^d$ → **Blackbox model** → Output $f(\boldsymbol{y}) \in \mathcal{V}$

# Main motivation

## Parametric PDE

A parametric PDE takes the form

$$\mathcal{L}_{\boldsymbol{y}}[u(\cdot, \boldsymbol{y})] = 0$$

with suitable boundary conditions.

- Parametric variables $\boldsymbol{y} \in \mathcal{U}$.
- Physical variables $\boldsymbol{x} \in \Omega$.
- $\mathcal{L}_{\boldsymbol{y}}$ is an operator depending on the parameters $\boldsymbol{y}$ (e.g. differentiation wrt $\boldsymbol{x}$).
- $u(\cdot, \boldsymbol{y})$ is an element of some Banach or Hilbert space $\mathcal{V}$.

Example:

$$-\nabla_{\boldsymbol{x}} \cdot (a(\boldsymbol{x}, \boldsymbol{y}) \nabla_{\boldsymbol{x}} u(\boldsymbol{x}, \boldsymbol{y})) = g(\boldsymbol{x}) \quad \text{in} \quad \Omega,$$

and BC.

## Main challenges

1. **High-dimensional models**: Often $d \gg 1$ or even $d = \infty$.

2. **The space $\mathcal{V}$ is infinite dimensional (Hilbert or Banach):**
   Needs discretization $\mathcal{V}_h$ over $\Omega$ $\rightsquigarrow$ induces a discretization error.

3. **Corrupted data (unknown errors):**
   Modelling errors, numerical error, random noise in the measurements.

4. **Generating data is expensive:**
   Example: generating multiple solutions of a particular PDE using a **black-box** numerical PDE solver.

# Holomorphy assumption

For $d \geq 1$, let $\boldsymbol{\rho} \in \mathbb{R}^d$ with $\boldsymbol{\rho} > 1$. The Bernstein polyellipse of polyradius $\boldsymbol{\rho}$ is

$$\mathcal{E}_{\boldsymbol{\rho}} = \mathcal{E}_{\rho_1} \times \mathcal{E}_{\rho_2} \times \cdots \mathcal{E}_{\rho_d} \subset \mathbb{C}^d.$$

where

$$\mathcal{E}_{\rho} = \{\frac{1}{2}(z + z^{-1}) : z \in \mathbb{C}, 1 \leq |z| \leq \rho\} \subset \mathbb{C}$$

## Assumption

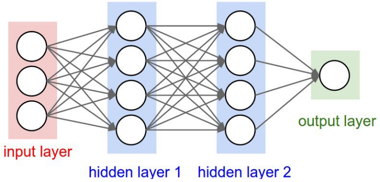
The function $f$ has a holomorphic extension from $[-1,1]^d$ to some Bernstein polyellipse $\mathcal{E}_{\boldsymbol{\rho}}$.

- Many parametric DEs provably satisfy this assumption, including elliptic diffusion equations, parametric IVPs,...

Cohen, DeVore, Schwab (2010, 2011), Chkifa, Cohen, Schwab (2015), Hoang, Schwab (2013, 2014)

# Outline

## Deep Neural Network (DNN) $\Phi : \mathbb{R}^d \to \mathbb{R}^K$




$$
\begin{aligned}
\boldsymbol{z}^{(1)} &= \sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{y} + \boldsymbol{b}^{(1)}\right), \\
\boldsymbol{z}^{(\ell)} &= \sigma\left(\boldsymbol{W}^{(\ell)}\boldsymbol{z}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\right), \qquad \ell = 2, \ldots, L-1 \\
\Phi(\boldsymbol{y}) &= \boldsymbol{W}^{(L)}\boldsymbol{z}^{(L-1)} + \boldsymbol{b}^{(L)}.
\end{aligned}
$$

- $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ are the **weights**.
- $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{N_\ell}$ are the **biases**.
- $\sigma$ is the **activation** function, e.g., $\sigma(t) = \max\{0, t\}$ (ReLU).

# Why DNN?

DNNs are capable of efficiently approximating functions from a wide variety of classes:

- Smooth functions, piecewise smooth functons, $H^k$ functions,...

[DeVore, Hanin, Petrova (2020)], [Elbrächter, Perekrestenko, Grohs, Bölcskei (2019)], and references therein.

- There are existence theorems about DNNs approximating holomorphic functions.
- These DNNs can achieve the same error bound as the best $s$-term polynomial approximation.
- Specifically, they can obtain an error proportional to $\exp\left(-\gamma s^{1/d}\right)$, where $\gamma$ depends on the region of holomorphy.
- The size and depth of these DNNs are bounded in terms of $s$ and $d$.

[Opschoor, Schwab, Zech (2019)], [Daws, Webster (2020)], [Adcock, Brugiapaglia, Dexter, Moraga (2021)].

## Input

- Each $d_i$ is uniquely represented as

$$d_i = f(\boldsymbol{y}_i) + n_i = \sum_{k=1}^{K} d_{i,k}\varphi_k \in \mathcal{V}_h, \qquad i = 1, \ldots, m.$$

- The values $\{(\boldsymbol{y}_i, d_i)\}_{i=1}^m$ are the *input*.

## Output

- Let $\{\Psi_i\}_{i=1}^N$ be a basis for $\mathcal{P}_\Lambda$, where $N = |\Lambda|$. Then we may write

$$\hat{f}_{\Lambda,h} : \boldsymbol{y} \mapsto \sum_{i=1}^{N} \left( \sum_{k=1}^{K} \hat{c}_{i,k}\varphi_k \right) \Psi_i(\boldsymbol{y}),$$

  where $\hat{c}_{i,k} \in \mathbb{R}$.
- The values $(\hat{c}_{i,k})_{n,k=1}^{N,K}$ are the *output*.

# Outline

## Practical DNN existence theorem for Hilbert-valued functions:

Let $f : \mathcal{U} \to \mathcal{V}$ be holomorphic in a suitable region, and $\widetilde{m} = cm/(\log^3(m)\log(d))$. Then there exists

1. a class of ReLU DNNs,
2. a loss function (regularized $\ell^2$-loss),
3. a choice of $m$ sample points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$,

such that any DNN $\Phi$ trained from the input $\{(\boldsymbol{y}_i, d_i)\}_{i=1}^m$ gives an approximation $f_\Phi$ satisfying

$$\|f - f_\Phi\|_{L^2_\varrho(\mathcal{U};\mathcal{V})} \lesssim (E_1 + E_2 + E_3),$$

$$E_1 = \exp\left(-\gamma \widetilde{m}^{1/(2d)}\right), \quad E_2 = \left(\frac{1}{m}\sum_{i=1}^m \|n_i\|_{\mathcal{V}}^2\right)^{1/2}, \quad E_3 = \|f - \mathcal{P}_h(f)\|_{L^\infty(\mathcal{U};\mathcal{V})}.$$

- $E_1$ is the approximation error: quantifies how well $f$ is approximated by a DNN in terms of $\widetilde{m}$.
- $E_2$ is the measurement error: quantifies the error in the pointwise evaluations of $f$ at the points $\boldsymbol{y}_i$.
- $E_3$ is the discretization error: since we work with $\mathcal{V}_h$ instead of $\mathcal{V}$.

Adcock, Dexter, Brugiapaglia and Moraga, Deep Neural Networks Are Effective At Learning High-Dimensional Hilbert-Valued Functions From Limited Data. MSML, volume 145, pages 1–36. (2021)

# Outline

## Orthogonal polynomials

- $\mathcal{U} = [-1,1]^d$ the unit hypercube.
- $\mathrm{d}\varrho(\boldsymbol{y}) = 2^{-d}\,\mathrm{d}\boldsymbol{y}$ be the uniform measure on $\mathcal{U}$.
- $\{\Psi_{\boldsymbol{\nu}}\}_{\boldsymbol{\nu}\in\mathbb{N}_0^d}$ be the tensor-product, orthonormal Legendre polynomial basis of $L^2_\varrho(\mathcal{U})$.

Let $L^2_\varrho(\mathcal{U};\mathcal{V})$ the Lebesgue-Bochner space of Hilbert-valued functions $f : \mathcal{U} \to \mathcal{V}$.

**Polynomial expansion:** if $f \in L^2_\varrho(\mathcal{U};\mathcal{V})$, then

$$f = \sum_{\boldsymbol{\nu}\in\mathbb{N}_0^d} c_{\boldsymbol{\nu}}\Psi_{\boldsymbol{\nu}}, \quad c_{\boldsymbol{\nu}} = \int_{\mathcal{U}} f(\boldsymbol{y})\Psi_{\boldsymbol{\nu}}(\boldsymbol{y})\,\mathrm{d}\varrho(\boldsymbol{y}) \in \mathcal{V}.$$

**Sequence in $\ell^p(\Lambda;\mathcal{V})$ :** For $1 \le p < \infty$ and $\boldsymbol{c} \in \ell^p(\Lambda;\mathcal{V})$, define

$$\|\boldsymbol{c}\|^p_{\mathcal{V},p} = \sum_{\boldsymbol{\nu}\in\Lambda} \|c_{\boldsymbol{\nu}}\|^p_{\mathcal{V}}.$$

# Polynomial approximation as a compressed sensing problem

Let $\Lambda$ be a finite index set with $|\Lambda| = N$. Define the normalized measurement matrix

$$\boldsymbol{A} = \left( \frac{\Psi_{\nu_j}(\boldsymbol{y}_i)}{\sqrt{m}} \right)_{i,j=1}^{m,N} \in \mathbb{R}^{m \times N},$$

and the normalized measurement and error vectors

$$\boldsymbol{b} = \frac{1}{\sqrt{m}} \left( f(\boldsymbol{y}_i) + n_i \right)_{i=1}^{m} \in \mathcal{V}_h^m, \quad \text{and} \quad \boldsymbol{e} = \frac{1}{\sqrt{m}} (n_i)_{i=1}^{m} \in \mathcal{V}^m.$$

Hence, the recovery of the polynomial coefficients $\boldsymbol{c}_\Lambda = (c_{\boldsymbol{\nu}})_{\boldsymbol{\nu} \in \Lambda}$ of $f$ is equivalent to solving the noisy linear system

$$\boldsymbol{A}\boldsymbol{c}_\Lambda + \boldsymbol{e} + \boldsymbol{e}' = \boldsymbol{b},$$

where

$$\boldsymbol{e}' = \frac{1}{\sqrt{m}} \left( f(\boldsymbol{y}_i) - f_\Lambda(\boldsymbol{y}_i) \right)_{i=1}^{m}.$$

Consider the Square Root LASSO problem

$$\min_{\boldsymbol{z} \in \mathcal{V}_h^N} \lambda \|\boldsymbol{z}\|_{\mathcal{V},1} + \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_{\mathcal{V},2}.$$

# Emulation as a DNN training problem

**Key insight: approximating polynomials as DNNs**

For any $\delta > 0$, there exists a DNN $\Gamma : \mathbb{R}^d \to \mathbb{R}^{|\Lambda|}$ (of size and depth depending on $d$, $|\Lambda|$ and $\delta$) such that

$$\|\Psi_{\boldsymbol{\nu}} - \Psi_{\boldsymbol{\nu},\delta}\|_{L^\infty(\mathcal{U})} \leq \delta,$$

where $\Gamma(\boldsymbol{y}) = (\Psi_{\boldsymbol{\nu},\delta}(\boldsymbol{y}))_{\boldsymbol{\nu}\in\Lambda}$.

[Opschoor, Schwab, Zech (2019)], [Daws, Webster (2020)], [Adcock, Brugiapaglia, Dexter, Moraga (2021)].

We can use this result to emulate the polynomial approximation problem as a DNN training problem:

$$\boldsymbol{A} = \left(\frac{\Psi_{\boldsymbol{\nu}_j}(\boldsymbol{y}_i)}{\sqrt{m}}\right)_{i,j=1}^{m,N} \in \mathbb{R}^{m\times N} \quad \rightsquigarrow \quad \boldsymbol{A}' = \left(\frac{\Psi_{\boldsymbol{\nu}_j,\delta}(\boldsymbol{y}_i)}{\sqrt{m}}\right)_{i,j=1}^{m,N} \in \mathbb{R}^{m\times N}$$

Carefully balancing the error due to this approximation and accounting for all other sources of errors leads to the main result.

# Outline

# Parametric PDE approximation

## A practical example:

- $\Omega = (0,1)^2$ physical domain with discretization $\Omega_h$.
- $\mathcal{U} = [-1,1]^d$ parametric domain with uniform probability measure.
- We seek a function $u : \Omega \times \mathcal{U} \to \mathbb{R}$ satisfying

$$-\nabla_{\boldsymbol{x}} \cdot (a(\boldsymbol{x},\boldsymbol{y})\nabla_{\boldsymbol{x}} u(\boldsymbol{x},\boldsymbol{y})) = g(\boldsymbol{x}) \quad \text{in} \quad \Omega, \quad \text{and BC.}$$

**Compute**: Approximation $u_{\Phi,h} : \mathcal{U} \to \mathcal{V}_h$ with a DNN $\Phi : \mathbb{R}^d \to \mathbb{R}^K$, of the form

$$u_{\Phi,h}(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{K} (\Phi(\boldsymbol{y}))_k \, \varphi_k(\boldsymbol{x}).$$

**Note:** we do not implement the training strategy from the theorem.

**Training:** Given data $\{(\boldsymbol{y}_i, d_i)\}_{i=1}^m$, $d_i = (c_k(\boldsymbol{y}_i))_{k=1}^K$ from a fixed FE discretization, minimize the loss function

$$\mathrm{MSE}(\boldsymbol{y}) := \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}(c_k(\boldsymbol{y}_i) - (\Phi(\boldsymbol{y}_i))_k)^2,$$

or

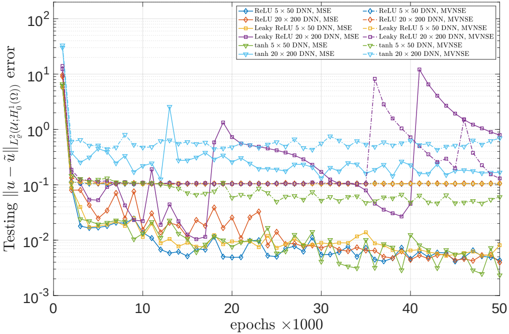
$$\mathrm{MVNSE}(\boldsymbol{y}) := \frac{1}{m}\sum_{i=1}^{m}\|u_h(\boldsymbol{y}_i) - u_{\Phi,h}(\boldsymbol{y}_i)\|_{\mathcal{V}}^2.$$

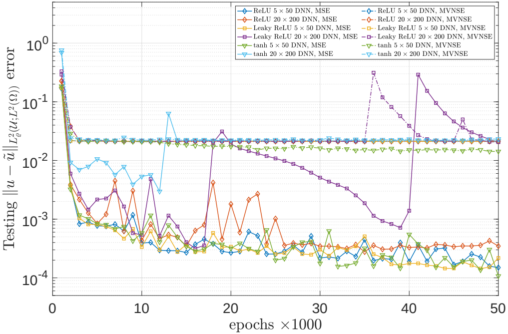
**Testing:** We compare the testing error in $L^2_\varrho(\mathcal{U}; L^2(\Omega))$ and $L^2_\varrho(\mathcal{U}; H^1_0(\Omega))$ norm.

- We use deterministic high-order sparse grid stochastic collocation method.

# Effective architectures and loss functions

- DNN architectures with MVNSE underperform identical architectures trained with the MSE.
- Big difference between in the $L^2(\Omega)$-norm (**right**) for tanh, ReLU and Leaky-ReLU $5 \times 50$ DNNs.

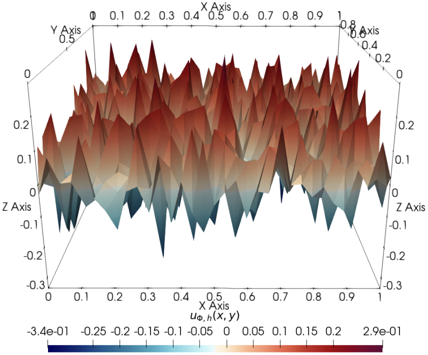


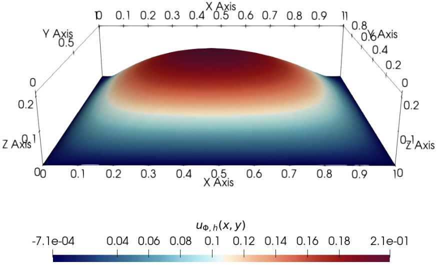



## Visualization comparative

Prediction for $u_h(\boldsymbol{x}, \boldsymbol{y})$ from a tanh $5 \times 50$ DNN at $\boldsymbol{y} = [0.995, 0]^t$

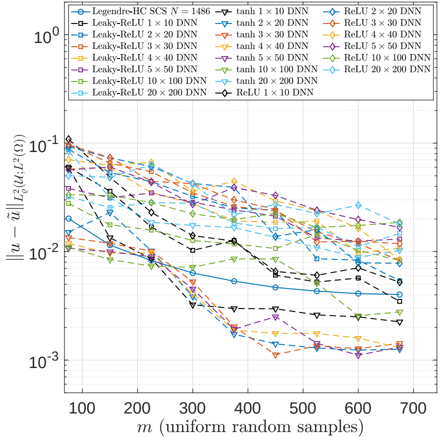
- Early training: after 2 epochs of Adam (MSE 6.4255).
- At the end of the training: after 2045 epochs (MSE $4.879 \cdot 10^{-7}$).

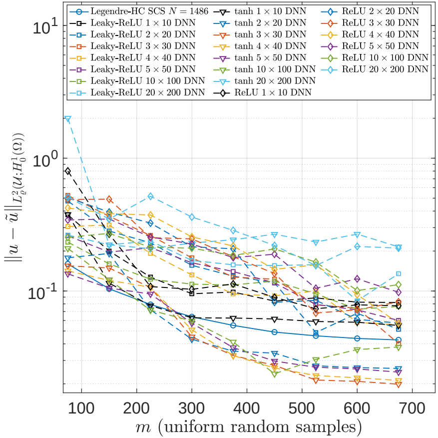


# Comparison with Simultaneous Compressed Sensing (SCS)

- Elliptic PDE with d=30 dimensional log-affine parametric diffusion.
- DNNs can outperform state-of-art polynomial-based CS methods.



# A mixed formulation

Define $\mathbb{K} = diag([a_1, a_2])$ and

$$\begin{aligned} -\nabla \cdot (\mathbb{K}(\boldsymbol{x}, \boldsymbol{y}) \nabla \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y})) &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) \quad && \boldsymbol{x} \in \Omega, \boldsymbol{y} \in \mathcal{U}, \\ \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y}) &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) \quad && \boldsymbol{x} \in \Gamma_D, \boldsymbol{y} \in \mathcal{U}, \\ \nabla \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{n} &= 0 \quad && \boldsymbol{x} \in \Gamma_N, \boldsymbol{y} \in \mathcal{U}. \end{aligned}$$

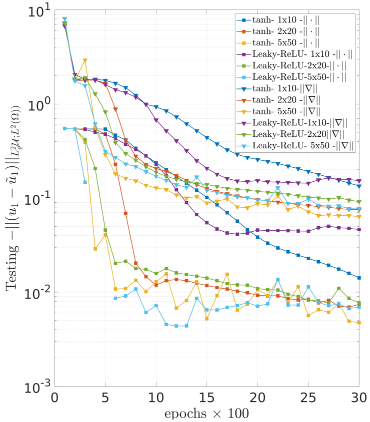
Given $\boldsymbol{y} \in \mathcal{U}$, find $(\boldsymbol{u}(\boldsymbol{y}), \boldsymbol{\sigma}(\boldsymbol{y})) \in [L^2(\Omega)] \times \boldsymbol{H}_{\Gamma_N}(\boldsymbol{div}; \boldsymbol{\Omega})$ such that

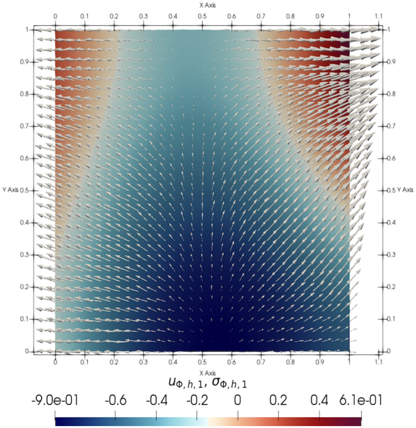
$$\begin{aligned} \langle \boldsymbol{\sigma}, \boldsymbol{\tau} \rangle_{L^2(\Omega)} + \langle \boldsymbol{u}, \nabla \cdot \boldsymbol{\tau} \rangle_{L^2(\Omega)} &= \langle \boldsymbol{\tau} \cdot \boldsymbol{n}, \boldsymbol{h} \rangle_{\Gamma_D} \\ \langle \nabla \mathbb{K} \cdot \boldsymbol{\sigma}, \boldsymbol{v} \rangle_{L^2(\Omega)} + \langle \mathbb{K} \boldsymbol{v}, \nabla \cdot \boldsymbol{\sigma} \rangle_{L^2(\Omega)} &= -\langle \boldsymbol{f}, \boldsymbol{v} \rangle_{L^2(\Omega)} \end{aligned}$$

Here $\sigma(\boldsymbol{y}) = \nabla \boldsymbol{u}(\boldsymbol{y}) \in \boldsymbol{H}_{\Gamma_N}(\boldsymbol{div}; \boldsymbol{\Omega})$.

# Parametric PDE with mixed B.C. example

- Testing errors for $\boldsymbol{u}$ are substantially smaller than those for its gradient $\nabla \boldsymbol{u}$.
- DNNs can be used as well to approximate parametric PDEs with mixed boundary conditions.



# Conclusions

- Deep learning is capable of approximating Hilbert-valued functions from limited data.
- There exists a DNN architecture and training procedure that performs as well as current best-in-class schemes.
- DNN can be used to approximate mixed formulations.
- Using the MSE loss function leads to better and faster approximations.
- In practice DNNs can outperform or match best current methods.

# References

B. Adcock, S. Brugiapaglia, N.Dexter, S. Moraga, Deep Neural Networks Are Effective At Learning High-Dimensional Hilbert-Valued Functions From Limited Data. MSML, volume 145, pages 1–36. (2021)

B. Adcock, S. Brugiapaglia, N.Dexter, S. Moraga, An efficient algorithm for computing near-optimal polynomial approximations of high-dimensional Hilbert-valued functions, in preparation (2021).

B. Adcock and N.Dexter, The gap between theory and practice in function approximation with deep neural networks. SIAM Journal on Mathematics of Data Science, 3(2), 624–655.

J. A. A. Opschoor, Ch. Schwab, and J. Zech, Exponential ReLU DNN expression of holomorphic maps in high dimension. SAM Research Report, 2019-35(35), 2019.

A. Chkifa, N. Dexter, H. Tran, and C. G. Webster., Polynomial approximation via compressed sensing of high-dimensional functions on lower sets. Math. Comp., 87(311):1415–1450, 2018.

**smoragas@sfu.ca**
**sites.google.com/view/sebanthalas**