Identifying the Higgs boson production in the $t\bar{t}H(b\bar{b})$ channel using quantum classifier models

Vasilis Belis (ETH Zürich)
Joint Annual Meeting of the Austrian Physical Society and Swiss Physical Society 2021
30 August - 3 September 2021, Universität Innsbruck

September 1, 2021
The $t\bar{t}H(b\bar{b})$ process

LO Feynman diagram of the **signal** and the dominant **background** processes in the semi-leptonic channel.

$$n_{\text{features}} = 8 \times 7(\text{jets}) + 7(\text{lepton}) + 4(\text{MET}) = 67$$
The $t\bar{t}H(b\bar{b})$ process

LO Feynman diagram of the **signal** and the dominant **background** processes in the semi-leptonic channel.

$n^\text{features} = 8 \times 7(\text{jets}) + 7(\text{lepton}) + 4(\text{MET}) = 67$

Analysis methods for $t\bar{t}H(b\bar{b})$ utilizing most features:

- ML models: Boosted Decision Trees (BDT), Deep Neural Networks (NN) exploiting all input feature correlations [ATL20, CMS19].
- Define physics-inspired high-level variables ($m^2$, jet shape, angular differences, etc.).
- State-of-the-art approaches for $t\bar{t}H(b\bar{b})$: graph and attention networks, etc.
Classification with conventional methods

- Assess performance of realistic HEP approaches on our data set (Delphes simulation).
- Full CMS simulation yields higher classifier performance.
- Models trained on full set of input features (67) and a reduced set (16) → benchmark.
- Measure of information loss (discriminating power reduction).
Why quantum machine learning for HEP?

- Heuristic answer: investigate the new set of ML techniques and methods available and assess advantages.

- Fundamental motivation: can quantum models utilise the quantum correlations inherent in HEP data leading to performance advantages?

- Goal in "ML jargon" \( \text{KBS21} \): Find inductive bias based on prior knowledge on the data generation (quantum process for HEP data).

- If this bias can be constructed and is classically difficult to simulate \( \rightarrow \) quantum advantage.

- Example: quantum algorithm for HEP event shower simulation, produces accurate results \( \text{NPdJB21} \). Can simulate naturally the interference diagram.
Why quantum machine learning for HEP?

- Heuristic answer: investigate the new set of ML techniques and methods available and assess advantages.
- Fundamental *motivation*: can quantum models utilise the quantum correlations inherent in HEP data leading to performance advantages?
  - Goal in “ML jargon” [KBS21]: Find inductive bias based on prior knowledge on the data generation (*quantum* process for HEP data).
  - If this bias can be *constructed* and is *classically difficult* to simulate → *quantum advantage*. 
Motivation

Why quantum machine learning for HEP?

• Heuristic answer: investigate the new set of ML techniques and methods available and assess advantages.

• Fundamental motivation: can quantum models utilise the quantum correlations inherent in HEP data leading to performance advantages?
  • Goal in “ML jargon” [KBS21]: Find inductive bias based on prior knowledge on the data generation (quantum process for HEP data).
  • If this bias can be constructed and is classically difficult to simulate → quantum advantage.

• Example: quantum algorithm for HEP event shower simulation, produces accurate results [NPdJB21]. Can simulate naturally the interference diagram.
Hybrid Quantum-Classical machine learning models

- Noisy Intermediate Scale Quantum (NISQ) devices:
  - **Circuit width**: limited number of qubits (superconducting qubits at IBM ~ 50).
  - **Circuit depth**: limited number of operations per qubit (small decoherence times).
Hybrid Quantum-Classical machine learning models

• Noisy Intermediate Scale Quantum (NISQ) devices:
  • Circuit width: limited number of qubits (superconducting qubits at IBM ~ 50).
  • Circuit depth: limited number of operations per qubit (small decoherence times).

Quantum Machine learning models for classification:

• Kernel methods → Quantum Support Vector Machine (QSVM)
• Quantum “Neural Networks” → Variational/Parametrized Quantum Circuits (VQC/PQC)

→ To accommodate NISQ limitations feature reduction is needed.
1. AutoEncoders (AE)

- Two AutoEncoders: one with 16 latent space features and one with 8.

2. Feature Selection

- Select 16 (8) input variables with the highest discriminative power according to their AUC score (Area Under Receiver Operating Characteristic curve).
Support Vector Machines

The SVM objective function is equivalent to (dual Lagrangian):

\[
\max \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\vec{x}_i \cdot \vec{x}_j) y_j c_j
\]

subject to

\[
\sum_{i=1}^{n} c_i y_i = 0, \quad \text{and} \quad 0 \leq c_i \leq C \quad \text{for all} \quad i
\]

- Kernel trick:
  \[
  \vec{x}_i \cdot \vec{x}_j \Rightarrow k(\vec{x}_i, \vec{x}_j) : \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)
  \]

- Make the kernel quantum
Support Vector Machines

- SVM objective function is equivalent to (dual Lagrangian)

\[
\text{maximize } L(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\vec{x}_i \cdot \vec{x}_j) y_j c_j
\]

subject to \( \sum_{i=1}^{n} c_i y_i = 0 \), and \( 0 \leq c_i \leq C \) for all \( i \)

- Kernel trick:

\[
(\vec{x}_i \cdot \vec{x}_j) \mapsto k(\vec{x}_i, \vec{x}_j) := \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)
\]
Support Vector Machines

- SVM objective function is equivalent to (dual Lagrangian)

\[
\text{maximize } L(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\vec{x}_i \cdot \vec{x}_j) y_j c_j
\]

subject to \(\sum_{i=1}^{n} c_i y_i = 0\), and \(0 \leq c_i \leq C\) for all \(i\)

- Kernel trick:

\((\vec{x}_i \cdot \vec{x}_j) \mapsto k(\vec{x}_i, \vec{x}_j) := \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)\)

- Make the kernel quantum:

\[
|0\rangle \rightarrow U^\dagger(\vec{x}_i) \rightarrow U(\vec{x}_j) \rightarrow K_{ij} = |\langle 0|U^\dagger(\vec{x}_i)U(\vec{x}_j)|0\rangle|^2
\]
Variational Quantum Circuits

- Data embedding circuit (feature map) here is fixed.
- Layers of parametrised quantum gates $\rightarrow$ trainable parameters.

\[
\langle \mathcal{O} \rangle_{\vec{x},\vec{\theta}} > 0.5 \rightarrow \text{signal, otherwise background.}
\]
Variational Quantum Circuits

- Data embedding circuit (feature map) here is fixed.
- Layers of parametrised quantum gates → trainable parameters.
- Output of the model → expectation value of an observable on the prepared state $|\psi(x, \theta)\rangle$ e.g. measure the first qubit on the computational basis

$$\mathcal{O} = \sigma_z \otimes 1 \otimes \cdots \otimes 1,$$

$$f(x, \theta) = \langle \psi(x, \theta) | \mathcal{O} | \psi(x, \theta) \rangle \equiv \langle \psi(x) | G^\dagger(\theta) \mathcal{O} G(\theta) | \psi(x) \rangle \equiv \langle \mathcal{O} \rangle_{x, \theta}.$$
Variational Quantum Circuits

- Data embedding circuit (feature map) here is fixed.
- Layers of parametrised quantum gates → trainable parameters.
- Output of the model → expectation value of an observable on the prepared state $|\psi(\vec{x}, \vec{\theta})\rangle$ e.g. measure the first qubit on the computational basis $\sigma_z \otimes 1 \otimes \cdots \otimes 1$,

$$
f(\vec{x}, \vec{\theta}) := \langle \psi(\vec{x}, \vec{\theta}) | \mathcal{O} | \psi(\vec{x}, \vec{\theta}) \rangle \equiv \langle \psi(\vec{x}) | G^\dagger(\vec{\theta}) \mathcal{O} G(\vec{\theta}) | \psi(\vec{x}) \rangle \equiv \langle \mathcal{O} \rangle_{\vec{x}, \vec{\theta}}.
$$

- Classification: if $\langle \mathcal{O} \rangle_{\vec{x}, \vec{\theta}} > 0.5 \rightarrow$ signal, otherwise background.
QSVM results with reduced features

**AE latent features (16)**

- QSVM (4 qubits): AUC = 0.621 ± 0.031
- SVM rbf: AUC = 0.619 ± 0.024
- QSVM (8 qubits): AUC = 0.620 ± 0.032
- Random Classifier

**AUC-based input feature selection (16)**

- QSVM (4 qubits): AUC = 0.657 ± 0.014
- SVM rbf: AUC = 0.651 ± 0.010
- Random Classifier

---

$N_{\text{train}}=576, N_{\text{test}}=720$ (x5)
VQC results with feature selection

Data encoding for the VQC model [HCTea19]:

Parametrised quantum circuit ("QNN"): 

AUC-based input feature selection (8)
Summary

Investigated:

- Different quantum algorithms QSVM and VQC.
- Data encoding circuits (amplitude encoding, direct encoding and data re-uploading).
- Feature dimensionality reduction methods.
- Classical benchmarks against state-of-the-art approaches in HEP and ML.

Our results [BGR+21]:

- Classical and quantum models have similar performance for the challenging $t\bar{t}H(b\bar{b})$ classification task (in agreement with previous studies [TKK+21, BS20, WCG+20, MJV+17]).
- The feature reduction procedure is extremely crucial (high impact on model performance).
Outlook & ongoing work

- Hybrid quantum-classical Autoencoder-based feature reduction.
  - Novel architectures: Preserve/enhance classification power in the latent space.

- Implementation of the algorithms on NISQ devices.
  - Assess the effect of the different noise components on model performance.
  - Error mitigation protocol if needed.

- Anomaly detection studies for model independent searches in HEP.
Thank you!
Backup
Why is it important?

- Study the Yukawa couplings of the Higgs in a purely fermionic process
- $t\bar{t}H$ coupling carries direct information about the scale of new physics [BS15]

→ Both processes have identical final state
Monte Carlo simulation: generation with Powheg v.2, parton shower Pythia 8 and Delphes v.3.4.1 (CMS Run II settings)

- Nominally: $n_{\text{jets}} = 6$ and $n_{\text{b-jets}} = 4$
- Jet observables (8): $(p_T, \eta, \phi, E, \text{b-tag}, p_x, p_y, p_z)$
- Semi-leptonic channel to reduce QCD background
  $\rightarrow 1$ lepton and 1 neutrino (MET) per event
- MET observables (4): $(p_T, p_x, p_y, \phi)$
- Lepton observables (7): $(p_T, \eta, \phi, E, p_x, p_y, p_z)$
- Keep 7 most energetic jets per event allowing for 1 correction of final or initial state radiation

$\Rightarrow n^{\text{features}} = 8 \times 7(\text{jets}) + 7(\text{lepton}) + 4(\text{MET}) = 67$
Pre-processing and pre-selection

Object pre-selection:

- $p_T > 30 \text{ GeV}, |\eta| < 2.1$ and iso $> 0.1$ for the electrons
- $p_T > 26 \text{ GeV}, |\eta| < 2.1$ and iso $> 0.1$ for the muons
- $p_T > 30 \text{ GeV}, |\eta| < 2.4$ for the jets

Event selection:

\[ n^{\text{jet}} \geq 4, \quad n^{b\text{-tag}} \geq 2 \quad \text{and} \quad n^{\text{leptons}} = 1 \]

- \( b\text{-tag} \in \{0, 1, \ldots, 7\} \), for different efficiencies
  
  → redefinition: \( b\text{-tag}' = \begin{cases} 1, & \text{if } b\text{-tag} > 1 \\ 0, & \text{otherwise} \end{cases} \)
Goal: Preserve non-linear correlations in the latent representation space

- Developed two models: 8 and 16-dimensional latent space
- Input features normalised to [0, 1] (min-max scaling)

\[ x_i \rightarrow \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \]

Model Architecture:
- Fully connected feed forward layers
- ELU activation functions. Sigmoid on latent and output layers
## Auto-Encoder hyperparameters

<table>
<thead>
<tr>
<th></th>
<th>PyTorch AE</th>
<th>TensorFlow AE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Layer Type</strong></td>
<td>Dense</td>
<td></td>
</tr>
<tr>
<td><strong>Encoder hidden layers</strong></td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td><strong>Latent space dim.</strong></td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td><strong>Loss</strong></td>
<td>Mean Square Error (MSE)</td>
<td></td>
</tr>
<tr>
<td><strong>Optimizer</strong></td>
<td>Adam</td>
<td></td>
</tr>
<tr>
<td><strong>Learning Rate</strong></td>
<td>$2 \times 10^{-3}$</td>
<td>$\sqrt{3} \times 10^{-3}$</td>
</tr>
<tr>
<td><strong>Batch size</strong></td>
<td>128</td>
<td>93</td>
</tr>
<tr>
<td><strong>Number of epochs</strong></td>
<td>80</td>
<td>30</td>
</tr>
</tbody>
</table>
Auto-Encoder training

\[ L(\theta) = \frac{1}{N} \sum_{i=0}^{N} |\vec{x}^i - \vec{x}^i_\theta|^2 \]

- Data set split 80%/10%/10% (train/validate/test):
  \[ N^{\text{train}} = 1.1 \times 10^6 \]
  \[ N^{\text{test}} = N^{\text{valid.}} = 1.44 \times 10^5 \]
- Compute validation loss after each epoch (probe for over-training)
  \[ L^{\text{test}} = 6.41 \times 10^{-4} \]
Reconstruction of the features

(d) PyTorch Auto-Encoder (16)

(e) TensorFlow Auto-Encoder (8)
The qubit:

\[ |\psi\rangle = \alpha |0\rangle + \beta |1\rangle \]
The qubit:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \equiv \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle$$

Generic qubit operations (quantum gates)

$$U = e^{-i\vec{\theta} \cdot \vec{\sigma}} \in SU(2):$$

$$U(\theta, \phi, \lambda) = \begin{pmatrix}
\cos\left(\frac{\theta}{2}\right) & -e^{i\lambda} \sin\left(\frac{\theta}{2}\right) \\
e^{i\phi} \sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)} \cos\left(\frac{\theta}{2}\right)
\end{pmatrix}$$
The qubit:

\[ |\psi\rangle = \alpha |0\rangle + \beta |1\rangle \equiv \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle \]

Generic qubit operations (quantum gates)

\[ U = e^{-i \hat{\theta} \cdot \hat{\sigma}} \in SU(2): \]

\[ U(\theta, \phi, \lambda) = \begin{pmatrix}
\cos\left(\frac{\theta}{2}\right) & -e^{i\lambda} \sin\left(\frac{\theta}{2}\right) \\
e^{i\phi} \sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)} \cos\left(\frac{\theta}{2}\right)
\end{pmatrix} \]

Construct all possible gates from \( U(\theta, \phi, \lambda) \)

\[ H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \equiv U\left(\frac{\pi}{2}, 0, \pi\right) \]

\[ |\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \]
Quantum gates and universality

Single qubit gates:
• A generic quantum gate can be decomposed in a series of $R_y$ and $R_z$ [BBC+95]

$$U(\theta, \phi, \lambda) = R_z(\lambda)R_y(\theta)R_z(\phi)$$

• For hardware implementation: more convenient to decompose to gates that have a direct physical operation analogue on the device.
Quantum gates and universality

Single qubit gates:
- A generic quantum gate can be decomposed in a series of $R_y$ and $R_z$ [BBC+95]

$$U(\theta, \phi, \lambda) = R_z(\lambda)R_y(\theta)R_z(\phi)$$

- For hardware implementation: more convenient to decompose to gates that have a direct physical operation analogue on the device.

Multi-qubit gates:
- 2-qubit SWAP and CNOT (Control-X) gates and the 3-qubit Toffolli gate

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- Any control-$U$ gate can be written as a combination of $CX$, $R_y$ and $R_z$ gates.
Quantum gates and universality

Single qubit gates:
- A generic quantum gate can be decomposed in a series of $R_y$ and $R_z$ [BBC+95]

$$U(\theta, \phi, \lambda) = R_z(\lambda)R_y(\theta)R_z(\phi)$$
- For hardware implementation: more convenient to decompose to gates that have a direct physical operation analogue on the device.

Quantum Gate Universality [DiV95]: The above “building blocks” can construct any quantum circuit acting on $n$ qubits, i.e. $SU(2^n)$, operating on at most two-qubits at a time.

Multi-qubit gates:
- 2-qubit SWAP and CNOT (Control-X) gates and the 3-qubit Toffolli gate

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
- Any control-$U$ gate can be written as a combination of $CX$, $R_y$ and $R_z$ gates.
Quantum classifiers

Kernel-based models (Quantum Support Vector Machines):
- Convex optimization tasks
- $\mathcal{O}(n^2)$ complexity construction of the kernel matrix elements

Quantum Neural Networks (Variational Quantum Circuits):
- Non-convex optimization
- $\mathcal{O}(n)$ complexity

Encoding (embedding) the classical data in a quantum circuit:

$|\psi(x)\rangle = G(\vec{x}) |0\rangle \otimes n$ qubits

- Amplitude encoding: exponentially decrease the needed number of qubits but have deep circuits
- Angle (direct) encoding: map each feature to a separate qubit shallow but wider circuits

Data re-uploading:
Repeat any data embedding circuit
Quantum classifiers

Kernel-based models (Quantum Support Vector Machines):
- Convex optimization tasks
- $O(n^2)$ complexity construction of the kernel matrix elements

Quantum Neural Networks (Variational Quantum Circuits):
- Non-convex optimization
- $O(n)$ complexity

Encoding (embedding) the classical data in a quantum circuit [SP18]:

$$|\psi(x)\rangle = G(\tilde{x}) |0\rangle \otimes n_{\text{qubits}}$$
Quantum classifiers

Kernel-based models (Quantum Support Vector Machines):
• Convex optimization tasks
• $\mathcal{O}(n^2)$ complexity construction of the kernel matrix elements

Quantum Neural Networks (Variational Quantum Circuits):
• Non-convex optimization
• $\mathcal{O}(n)$ complexity

Encoding (embedding) the classical data in a quantum circuit [SP18]:

$$|\psi(x)\rangle = G(\vec{x}) |0\rangle \otimes n_{\text{qubits}}$$

• Amplitude encoding: exponentially decrease the needed number of qubits but have deep circuits
• Angle (direct) encoding: map each feature to a separate qubit shallow but wider circuits
• Data re-uploading [PSCLGFL20]: repeat any data embedding circuit
Quantum Support Vector Machines

Sample the kernel matrix on a quantum device (multiple measurements)

Maximise the SVM objective function on a classical computer

Amplitude encoding circuit

\[ \phi: \mathbb{R}^N \rightarrow \mathcal{H} \otimes n \text{qubits} \Rightarrow \vec{x} \in \mathbb{R}^{16} \rightarrow |\psi_{\vec{x}}\rangle = \frac{1}{\sqrt{16}} \sum_{i=0}^{m} m_i |i\rangle, m_i \text{norm. inputs} \]

\[ K_{ij} = |\langle 0|U^\dagger(\vec{x}_i)U(\vec{x}_j)|0\rangle|^2 \]

\[ \begin{array}{c}
|0\rangle \\
|0\rangle \\
\vdots \\
|0\rangle \\
\end{array} \xrightarrow{U^\dagger(\vec{x}_i) \quad U(\vec{x}_j)} \Rightarrow K_{ij} = |\langle 0|U^\dagger(\vec{x}_i)U(\vec{x}_j)|0\rangle|^2 \]
Quantum Support Vector Machines

\[ K_{ij} = |\langle 0 | U^\dagger(\vec{x}_i) U(\vec{x}_j) |0 \rangle|^2 \]

Amplitude encoding circuit
\[ \phi : \mathbb{R}^N \rightarrow \mathcal{H} \otimes n \text{ qubits} \Rightarrow \vec{x} \in \mathbb{R}^{16} \rightarrow |\psi_{\vec{x}}\rangle = \frac{1}{4^{15}} \sum_{i=0}^{m} m_i |i\rangle, \text{ norm. inputs} \]
Quantum Support Vector Machines

- Sample the kernel matrix on a quantum device (multiple measurements)
- Maximise the SVM objective function on a classical computer

Amplitude encoding circuit

$$\phi : \mathbb{R}^N \rightarrow \mathcal{H}^{\otimes n_{\text{qubits}}} \Rightarrow \vec{x} \in \mathbb{R}^{16} \rightarrow |\psi_{\vec{x}}\rangle = \frac{1}{4} \sum_{i=0}^{15} m_i |i\rangle, \ m_i \text{ norm. inputs}$$
Alternative data encoding circuit (8-qubits)
QSVM results on the input space

\[ N^{\text{train}} = 576, \ N^{\text{test}} = 720 \ (x5) \]

- QSVM (6 qubits): AUC = 0.676 ± 0.017
- SVM linear: AUC = 0.672 ± 0.017
- Random Classifier

64 out of the 67 input features

“Realistic” approach

\[ N^{\text{(train)}}: \ 1827808, \ N^{\text{(test)}} 456952 \]

- DNN, AUC = 0.704 ± 0.001
- DNN(latent), AUC = 0.623 ± 0.002
- BDT, AUC = 0.691 ± 0.001
- BDT(latent), AUC = 0.652 ± 0.002
# QSVM feature reduction benchmarks

<table>
<thead>
<tr>
<th>Feature selection + Model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO + QSVM</td>
<td>0.66 ± 0.01</td>
</tr>
<tr>
<td>PyTorch AE + QSVM</td>
<td>0.62 ± 0.03</td>
</tr>
<tr>
<td>INFO + SVM rbf</td>
<td>0.65 ± 0.01</td>
</tr>
<tr>
<td>PyTorch AE + SVM rbf</td>
<td>0.62 ± 0.02</td>
</tr>
<tr>
<td>KMeans + SVM rbf</td>
<td>0.61 ± 0.02</td>
</tr>
</tbody>
</table>

(a) 16 input variables

<table>
<thead>
<tr>
<th>Feature selection + Model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO + QSVM</td>
<td>0.68 ± 0.02</td>
</tr>
<tr>
<td>INFO + Linear SVM</td>
<td>0.67 ± 0.02</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.68 ± 0.02</td>
</tr>
</tbody>
</table>

(b) 64 (QSVM, LSVM) and 67 (LR) input variables

- Trained and tested (same data set size) a collection of classical models (SVMs, Logistic Regression, BDT, Random Forests, Multilayer Perceptrons, kNN, Naive Bayes and QDA).
- Feature extraction techniques: PCA, K-means, Truncated SVD, Isomap and Locally Linear Embedding.
VQC benchmarks

<table>
<thead>
<tr>
<th>Feature selection + Model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO + VQC</td>
<td>0.66 ± 0.01</td>
</tr>
<tr>
<td>INFO + Random Forest</td>
<td>0.66 ± 0.02</td>
</tr>
<tr>
<td>KMeans + Log. Regr.</td>
<td>0.64 ± 0.01</td>
</tr>
<tr>
<td>TensorFlow AE + AdaBoost</td>
<td>0.63 ± 0.03</td>
</tr>
</tbody>
</table>

- Needs more training data to achieve same performance as QSVM.
- VQC poor performance with amp. enc. 16 features and 8 AE features (AUC~ 0.55) → resort to feature selection of 8 input features.


*Measurement of \( t\bar{t}H \) production in the \( H \rightarrow b\bar{b} \) decay channel in 41.5 fb\(^{-1} \) of proton-proton collision data at \( \sqrt{s} = 13 \) TeV*, Tech. Report CMS-PAS-HIG-18-030, CERN, Geneva, 2019.


Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre, *Data re-uploading for a universal quantum classifier*, Quantum **4** (2020), 226.
