# how to improve the results and by using which and explain it in detail using simple words with full text s00542 024 05776-y 1 .pdf and suggest some video tutorials URL

To improve prediction results when using a hardware-accelerated neural network model for early detection of cardiac events (like sudden cardiac arrest), hereʼs a simple, step-by-step explanation—along with detailed strategies and relevant video tutorials.

## How to Improve Prediction Results

1. **Optimize Neural Network Parameters**

 **Hyperparameter Tuning:** Adjust settings such as the number of layers, learning rate, batch size, and number of neurons. These parameters are crucial—choosing the right values can make the neural network more accurate in classifying patient conditions.

 *Example*: A higher number of layers can help the model learn more complex patterns, but too many can cause overfitting. Find the right balance by experimenting.

 **Activation Functions:** Use advanced functions like ReLU or Leaky ReLU. These help the network learn better patterns from the data 1  .

## Data Preprocessing

 **Feature Engineering:** Carefully select which aspects of the heart rate variability HRV) or ECG signals are sent to the model. Linear and nonlinear HRV features can boost prediction accuracy 2  .

 **Balancing Data:** If some conditions are much rarer than others (class imbalance), techniques like SMOTE or two-stage sampling SMOTEENN, SMOTETomek) can help balance the classes for better generalization 1  .

 **Noise Reduction:** Clean your data by removing artifacts, abnormal beats, or noise before training the network.

## Training Methods

 **Dropout Regularization:** Prevents overfitting by “turning off” parts of the network during training. This makes the network less reliant on any one path and improves real-world performance 1  .

 **Data Augmentation:** Use synthetic data or increase your training set size. Larger datasets allow the network to generalize better and avoid learning noise or rare patterns only.

## Model Selection

 **Choose the Right Network:** Convolutional Neural Networks CNNs are excellent for temporal data like ECG/HRV because they can identify important features automatically 2  .

 **Try Multiple Models:** Compare traditional machine learning models SVM, Random Forest) with deep learning CNN, ANN . Deep learning often outperforms classical methods for complex signals 2 3  .

## Continuous Testing and Validation

 **Cross-Validation:** Test the neural network on different data subsets, not just a single test set. This detects if youʼre overfitting or genuinely learning general patterns.

 **Update Models:** Retrain the network when you have new data or new types of patients to keep the prediction accuracy high.

## Simple Explanation of the Process

**Think of the neural network as a very smart doctor who gets better at diagnosis the more examples it sees. To improve its skills, you carefully pick the most relevant patient symptoms, adjust how it learns, show it examples from both sick and healthy patients (in equal amounts), and routinely test its knowledge using new cases.**

## Video Tutorials for Easy Understanding

### Neural Network for Heart Rate Variability and Classification YouTube)

 [Multi Class Stress Detection Through Heart Rate Variability](https://www.youtube.com/watch?v=i9T3AyBawUc) – This video shows practical steps to create and validate a heart rate variability classifier. It discusses evaluating different algorithms and using deep neural networks effectively 4  .

### Deep Learning for ECG/Heart Disease Prediction

 [Convolutional Networks for Heart Disease Prediction](https://www.youtube.com/watch?v=fGv6VmfGMLc) – Explains how neural networks can predict heart problems by analyzing ECG/HRV data.

### Understanding Neural Networks for Health Signals

 [HeartTrack: Convolutional Neural Network for Remote Video-Based Heart Rate](https://openaccess.thecvf.com/content_CVPRW_2020/papers/w19/Perepelkina_HeartTrack_Convolutional_Neural_Network_for_Remote_Video-Based_Heart_Rate_Monitoring_CVPRW_2020_paper.pdf) [Monitoring CVPRW 2020](https://openaccess.thecvf.com/content_CVPRW_2020/papers/w19/Perepelkina_HeartTrack_Convolutional_Neural_Network_for_Remote_Video-Based_Heart_Rate_Monitoring_CVPRW_2020_paper.pdf) – Though this is a paper, it contains visuals and simple explanations for processing health signals with CNNs.

## Summary Table: Tips to Improve Results

|  |  |  |
| --- | --- | --- |
| Step | What to Do | Why It Helps |
| Optimize Parameters | Adjust layers, learning rate, batch size | Makes training more effective and accurate |

|  |  |  |
| --- | --- | --- |
| Step | What to Do | Why It Helps |
| Clean and Balance Data | Use advanced preprocessing, balance classes | Prevents bias and boosts real-world usability |
| Choose Best Model | Use CNNs, compare with other techniques | Deep learning can capture complex ECG/HRV patterns best |
| Regularize & Augment | Use dropout, create synthetic data | Reduces overfitting, increases generalizability |
| Continuous Validation | Cross-validate, retrain with new data | Keeps model current and robust to new cases |

By following these steps and checking out the suggested tutorials, you can significantly boost the performance of heart-related neural network models—even without deep technical expertise 2 3 1  .

# how to improve the results and by using which and explain it in detail using simple words with full text s00542 024 05776-y 1 .pdf and suggest some video tutorials URL

I am currently unable to access the content of the file "s00542 024 05776-y 1 .pdf" directly. If you can provide the text or key excerpts from it, I can help explain how to improve the results mentioned in that document in simple terms.

Meanwhile, I can give you a general approach on how to improve performance for models such as those involving hardware-accelerated neural networks or early predictions in medical data, which are often topics in such papers:

## General Ways to Improve Results in Predictive Models Like Neural Networks

### Data Quality and Quantity

 Use clean, well-prepared data with minimal noise or errors.

 Increase the dataset size if possible—more data helps the model learn better and generalize.

### Feature Engineering

 Choose the most relevant features or inputs that impact the prediction.  Extract meaningful patterns and remove redundant information.

### Model Architecture Improvement

 Use more advanced network architectures (e.g., deeper layers or specialized models like CNNs or LSTMs).

 Experiment with different hyperparameters like learning rate, number of hidden units, activation functions.

### Regularization Techniques

 Apply techniques like dropout or weight decay to prevent overfitting.  Helps the model to generalize well on unseen data.

### Optimization Algorithms

 Use better optimizers like Adam, RMSProp, or learning rate schedules to help the model converge faster and better.

### Hardware Acceleration

 Utilize GPUs or specialized hardware FPGAs, ASICs) to speed up training and allow for more complex models.

### Cross-Validation and Testing

 Use robust validation schemes to avoid overfitting.  Test on varied datasets to ensure model reliability.

## Suggested Video Tutorials for Further Learning

 **Understanding Neural Networks and Improving Model Performance:** [Deep Learning Fundamentals - By deeplizard](https://www.youtube.com/watch?v=aircAruvnKk)

### Feature Engineering and Data Preparation:

[Feature Engineering for Machine Learning - by Simplilearn](https://www.youtube.com/watch?v=-_sUPF3ixyM)

 **Hyperparameter Tuning Techniques Explained:** [Hyperparameter Tuning in Machine Learning - by StatQuest](https://www.youtube.com/watch?v=QgvwWsouIhE)

 **Using Hardware Acceleration GPUs for training Deep Learning Models:** [Deep Learning with GPUs - by NVIDIA Developer](https://www.youtube.com/watch?v=lv2XVvjH1lg)

### Improving Neural Network Performance with Regularization:

[Regularization & Dropout in Neural Networks - by Andrew Ngʼs Coursera excerpt](https://www.youtube.com/watch?v=ArPaAX_PhIs)

If you can share more details or relevant sections of the PDF, I will be happy to provide a detailed explanation tailored specifically to that paper.

# how to improve the results and by using which and explain it in detail using simple words with full text s00542 024 05776-y .pdf and suggest some video tutorials URL

Hereʼs a detailed and simple explanation, based directly on the content of **s00542 024 05776 y.pdf**, of how to improve the results of a hardware-accelerated neural network model for early prediction of sudden cardiac arrest SCA , and the methods that work best. Video tutorial recommendations are included at the end.

## How Were Good Results Achieved in the Paper?

### Better Features for Input Feature Engineering)

 The model uses *many heart rate variability HRV) features* extracted from ECG signals: time-domain SDRR, RMSSD, pNN20, pNN50 , frequency-domain LF, HF, LF/HF , and non-linear SD1, SD2, SD1/SD2 .

 Each feature brings extra useful information—for example, RMSSD reflects quick changes, while LF/HF shows the balance between different rhythms 42  .

 **How to improve further:** Try new HRV features, use data from other vital signs (like blood pressure), or employ methods to automatically select the most informative features.

### Neural Network Design

 The paper tested neural networks with different numbers of layers and neurons. Too few layers canʼt learn complex patterns; too many can "overfit" (do well on old data but poorly on new data).

 They found that **5 layers 10 16 32 64 1 nodes)** worked best: simple enough to avoid overfitting, but complex enough to capture patterns.

 *Dropout* and *regularization* were used to prevent overfitting—these are techniques that make the network more “cautious” so it doesnʼt memorize the training data.

 **How to improve further:** Experiment with networks having slightly different numbers of layers/neurons; try other architectures like 1D convolutional networks CNNs specialized for time-series.

### Cross-Validation Techniques

 The study used *leave-one-subject-out* and *k-fold cross-validation* for reliable evaluation. This helps ensure the model works for new, unseen patients.

 **How to improve further:** Add data from multiple hospitals or different populations to make the model robust for a variety of conditions.

### Hardware Acceleration FPGA Implementation)

 The model is implemented on an FPGA chip, making inference (prediction) about 500 times faster than regular software. This is crucial for wearable devices and real-time detection.

 Different precision levels were compared 16-bit, 18-bit, 20-bit). *18 -bit* was found to offer the best balance between speed, accuracy, and required hardware resources (cost-effective and efficient) 42 .

 **How to improve further:** For even faster or more power-efficient systems, try custom ASIC design or optimize further for memory/resource use if deploying on smaller/cheaper chips.

### Model Performance

 The neural network achieved **96.23% accuracy** for SCA prediction (higher than several previous research works compared in the paper).

## Summary Table: How to Improve the Results

|  |  |  |
| --- | --- | --- |
| Step | What They Did/What You Can Do Further | Why It Helps |
| Feature Engineering | Use a mix of time, frequency, non-linear metrics | Captures all patterns in heart signals |
| Model Design | 5-layer MLP, regularization, dropout | Balances learning and prevents overfitting |
| Validation | Leave-one-subject-out, k-fold cross- validation | Ensures model works for all, not just some |
| Hardware Optimization | FPGA, 18-bit precision | Enables instant, real-time predictions |
| Dataset Expansion | Try more subjects/diverse databases | Makes the model robust and reliable |

**Step-by-Step Improvements Explained Simply**

**Use More Diverse “Clues”:** Add different kinds of information from ECG to help the model spot subtle danger signs.

**Build the “Just-Right” Model:** Not too simple and not too complex—enough layers and neurons to learn patterns, and use dropout so it doesnʼt just memorize.

**Test the Model Many Ways:** Split the patient data in different ways to check if the model stays accurate on brand new patients.

**Make Predictions Super Fast:** Use specialized chips (like FPGA) so alerts are instant. Pick the right number of bits so itʼs accurate but still fits in cheap wearables.

**Keep Practicing with New Data:** If possible, retrain or fine-tune as more data comes in from different hospitals or populations.

## Recommended Video Tutorials

### Heart Rate Variability Analysis with Machine Learning

[Heart Rate Variability HRV Analysis - Real-time using Python & ML](https://www.youtube.com/watch?v=USGnvW6f3yo)

### Building and Tuning Neural Networks in Python

[Deep Learning with Keras - Neural Network Optimization](https://www.youtube.com/watch?v=tPYj3fFJGjk)

 **FPGA for AI/ML How hardware acceleration is done)** [AI & Neural Networks on FPGA FPGA4Student](https://www.youtube.com/watch?v=En1jzRboO7g)

 **Understanding Dropout and Regularization (for overfitting)** [What is Dropout in Neural Networks? StatQuest](https://www.youtube.com/watch?v=ARq74QuavAo)

### In summary:

To improve results, use diverse reliable features, carefully tune your neural network (not too simple, not too complex), validate carefully, optimize for hardware, and use cross-database data for generalizability. Hardware acceleration lets you achieve fast, wearable-ready deployment— key for tasks like early cardiac arrest prediction.

⁂

# how to improve the results and by using which and explain it in detail using simple words with full text s00542 024 05776-y .pdf and suggest some video tutorials URL

Based on the technical paper you shared (“Hardware-accelerated neural network model for early prediction of sudden cardiac arrest based on heart rate variability metrics”), hereʼs a simple, step-by-step explanation of **how to improve the results** of such a model, which methods to use, and why they work. Relevant video tutorials are also suggested for each step.

## How to Improve Results Based on the Paper)

1. **Use Better and More Features from HRV**

### What They Did:

The model used a wide range of features from heart rate variability HRV – time-domain

SDRR, RMSSD, pNN20, pNN50 , frequency-domain LF, HF, LF/HF ratio), and even non- linear features SD1, SD2, SD1/SD2 43 .

### How to Improve:

 Include all 10 HRV metrics as inputs, not just a few. Each captures different heart rhythm patterns.

 Rank features by importance (mutual information or similar) and focus on those that matter most.

 Try newly discovered or domain-specific features if available.

## Tune the Neural Network Architecture

### What They Did:

Multiple neural network NN) layouts were tested — for example, 4, 5, and 6 layers. The best was “10 16 32 64 1” (one input, three hidden, one output) 43 .

### How to Improve:

 Experiment with the number of layers and neurons. Too few = underfitting; too many = overfitting.

 Use “powers of two” for neurons (like 16, 32, 64 .

 Regularization (adding penalties for very large weights) and dropout (randomly turn off neurons during training) help prevent overfitting.

 Use the ReLU activation function in hidden layers for faster and deeper learning.

## Robust Evaluation Methods

### What They Did:

Used **leave-one-subject-out** and k-fold cross-validation, making sure the model works well for new patients, not just known ones 43  .

### How to Improve:

 Stick to strict cross-validation.

 Test on fresh, unseen subjects or populations if available.  Use multiple databases for variety.

## Hardware Optimization for Real-Time, Practical Use

### What They Did:

Implemented the NN on an FPGA (a type of chip), using fixed-point precision 18 bits), making it about **500 times faster** than regular software 43  .

### How to Improve:

 Choose the “right” hardware precision: the study found 18 bits to be perfectly balanced for accuracy and efficiency.

 Future: Try more efficient hardware like ASIC, or even optimize memory/logic further for wearables.

## Compare and Benchmark Against Other Methods

 The modelʼs performance 96.23% accuracy) was higher than similar works using other machine learning or neural network methods, because of the combination of richer features, optimized NN design, and hardware speed  43  .

## In Simple Words

**Give the Model More Useful Clues:** Use all kinds of HRV features, not just the basic ones.

**Choose the Best “Brain” NN Structure):** Not too simple, not too complex. Find the right number of hidden layers and neurons.

**Test Fairly and Strictly:** Use smart methods to check the model really works for new people.

**Speed It Up With Special Chips:** Make the model practical for real-world, instant prediction by putting it on an FPGA or similar chip.

**Constantly Compare:** Watch how your approach stacks up against earlier methods.

## Useful Video Tutorials

 **HRV Feature Extraction and ECG Analysis:** [Heart Rate Variability HRV Analysis Explained](https://www.youtube.com/watch?v=USGnvW6f3yo)

### Neural Network Tuning and Overfitting/Underfitting:

[Neural Networks: Overfitting, Underfitting, and Model Capacity Simplilearn)](https://www.youtube.com/watch?v=fUu8tuVb8D4)

### FPGA and Hardware Acceleration for AI

[FPGA for Artificial Intelligence Applications - FPGA4student](https://www.youtube.com/watch?v=En1jzRboO7g)

 **Dropout in Deep Learning (to prevent overfitting):** [Dropout in Neural Networks - StatQuest](https://www.youtube.com/watch?v=ARq74QuavAo)

### Summary Table

|  |  |
| --- | --- |
| What to Do | Why It Helps |
| Use diverse HRV features | Covers more heart signal patterns |
| Optimize NN design | Balances learning vs. over/underfitting |
| Strict cross-validation | Ensures model works for all, not just a few |
| Use hardware acceleration | Makes device fast and practical to use |
| Benchmark regularly | Confirms your model is truly an improvement |

**By following these steps, you can achieve high accuracy, practical speed, and strong reliability in predicting sudden cardiac arrest—directly following and expanding what worked best in the referenced technical paper.**

⁂

# send me the project tutorials based on this in vivado

Here are high-quality project tutorials and step-by-step guides on implementing neural networks and related machine learning architectures on FPGA using Vivado, as relevant to your referenced paper on hardware-accelerated neural networks for early cardiac event prediction:

## YouTube Video Tutorials

1. **Vivado Verilog Implementation of Neural Network**

**Overview:** Complete hardware implementation of a neural network on FPGA using Verilog in Vivado.

**Highlights:** Covers code writing, simulation, synthesis, and testing in Vivado. **For Beginners and Intermediate Users.**

### Watch here:

[Vivado Verilog implementation of Neural Network YouTube)](https://www.youtube.com/watch?v=NoSPWfOh1GU)  44

## Neural Networks on FPGA YouTube Series)

 **Overview:** Detailed introduction and design for implementing feedforward neural networks

FCNN) on FPGAs, using HDL and Vivado.

 **Part 1** Overview of neural network on FPGA [Watch here](https://www.youtube.com/watch?v=rw_JITpbh3k) 45

 **Part 2** Actual hardware neuron design (code walkthrough, memory for weights, interfacing) [Watch here](https://www.youtube.com/watch?v=a2wOjxRf_xg) 46

## How to Build a Neural Network on an FPGA

 **Platform:** PYNQ Z2 (can adapt concepts for Xilinx boards)  **Key Aspects:**

 Model training

 Optimizing and converting neural network for FPGA  Deploying on FPGA with Vivado/HLS flow

### Watch here:

[How to Build a Neural Network on an FPGA YouTube)](https://www.youtube.com/watch?v=3qtMs5jD-OY)  47

## FPGA Design Tutorial Vivado & Algorithm Implementation)

 **Walkthrough:** Project setup in Vivado, algorithm implementation, hardware testing.  **Watch here:**

[FPGA Design Tutorial: Vivado & HA Algorithm ... YouTube)](https://www.youtube.com/watch?v=jStQ3dX4Ihk)  48

## Written and PDF Tutorials

1. **Implementation of a Neural Network in FPGA VHDL, Vivado)**

 **Format:** Step-by-step thesis PDF  **Whatʼs Inside:**

 Explanation of CNN design and FPGA suitability  VHDL/Vivado workflow with diagrams

 Simulation using Vivado simulator

 Tips on resource optimization and validation  Great for serious project development

### View PDF

[Implementation of a Neural Network in FPGA Vivado](https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG_Andreu_Gim%C3%A9nez_v2.pdf?sequence=3&isAllowed=y) 49

## Blog Tutorial: Make Your Neural Network Hardware Accelerator

 **Walkthrough:** Linear Regression to neural net accelerator using PyTorch, Vivado, and PYNQ.

 **Covers:** Python model, HLS, Vivado integration, hardware deployment.

### Read here:

[Make Your Neural Network Hardware Accelerator Part-1](https://blog.devgenius.io/make-your-neural-network-hardware-accelerator-part-1-19cafdf24904)  50

## Additional Resources

### Open-source machine learning to FPGA (hls4ml):

[GitHub: fastmachinelearning/hls4ml](https://github.com/fastmachinelearning/hls4ml) – Provides HLS code and Vivado/HLS examples for deploying neural networks on FPGA, usable with Vivado 51  .

### Article: Building Neural Networks on FPGAs:

[Read here](https://resources.altium.com/p/building-neural-networks-fpgas) – Overview of workflow and key Vivado tasks for neural network projects 52  .

## Tips for Vivado Projects Based on the Paper

 **Start Simple:** Begin by implementing a simple MLP (multilayer perceptron) as shown in the paper.

 **Use Fixed-Point Arithmetic:** For efficiency, use 16- or 18-bit fixed-point as described in your reference. Vivado and Xilinx IP blocks support this natively.

 **Regularization and Dropout:** Implement dropout and weight regularization during software training, then transfer trained weights to FPGA for inference.

 **Testing:** Use Vivado simulator for initial debugging, then synthesize and deploy on Artix-7 or your available FPGA.

### These resources will guide you through designing, coding, simulating, and running neural networks on FPGA with Vivado, closely matching the approach detailed in the technical paper you referenced.

⁂

# how to improve the results and by using which and explain it in detail using simple words with full text s00542 024 05776-y .pdf and suggest some video tutorials URL and send me the project tutorials based on this in vivado each separately

Hereʼs a detailed, easy-to-understand guide—using the main points from the PDF "Hardware- accelerated neural network model for early prediction of sudden cardiac arrest based on heart rate variability metrics"—on 1) how to improve classification results, 2) which methods and tools to use, and 3) project tutorials for Vivado FPGA design based on this work. **Video**

**tutorials** are included at the end, with separate resources for both machine learning and Vivado hardware implementation.

## How to Improve the Results: Step-by-Step, in Simple Words

* 1. **Use the Best HRV Features**

 **What this means:** The model analyses the heart's electrical signal (from ECG) to extract "clues" called heart rate variability HRV) features, such as SDRR, RMSSD, pNN20, pNN50, LF, HF, LF/HF, SD1, SD2, and SD1/SD2.

 **Why it matters:** More and better features mean the model has more information to spot warning signs of sudden cardiac arrest SCA .

 **How to improve:** Use all available HRV metrics (time, frequency, and non-linear features). Consider ranking them (the authors used "mutual information" to measure which features mattered most).

## Pick the Optimal Neural Network Design

 **What this means:** The neural network NN) is the “smart brain” that learns to tell apart normal vs. SCA ECG segments.

 **What worked best in the paper:** A multilayer perceptron MLP) with 5 layers: [input:10 – [hidden: 16, 32, 64 – [output:1 , each layer using more neurons (powers of two) for more complex learning but not so many as to "overfit."

### How to improve:

 Tune the number of layers and neurons.

 Use activation functions like ReLU in hidden layers and Sigmoid at output.

 Use dropout (randomly ignore some neurons during training) and regularization (penalize big weights) to avoid overfitting.

## Evaluate the Model the Right Way

 **What this means:** Test your modelʼs accuracy on data from new patients, not just those it was trained on.

 **How:** Use "leave-one-subject-out" and k-fold cross-validation. This ensures your model is reliable for everyone, not just those like your training set.

## Implement on Hardware for Real-Time Prediction

 **What this means:** To make predictions instantly and run on wearable devices, the model was programmed into an FPGA chip using Verilog in Vivado.

### How to improve:

 Use fixed-point arithmetic (not floating point): 18-bit fixed-point was the sweet spot (almost as accurate as software, but much faster and fits into small chips).

 Optimize model structure (fewer layers if resources are tight) and test in simulation.

## Regularly Benchmark and Compare

 **What this means:** Always check if your system is better than previous ones (compare accuracy, sensitivity, specificity, speed, and hardware size).

## Video Tutorials — Each Topic Separately

* 1. **HRV Feature Extraction & ML Model Building**

 **Heart Rate Variability HRV Analysis Explained (step-by-step):** <https://www.youtube.com/watch?v=USGnvW6f3yo>

 **Neural Networks: Overfitting, Underfitting, and Model Tuning:** <https://www.youtube.com/watch?v=fUu8tuVb8D4>

 **Deep Learning with Keras - Neural Network Optimization:** <https://www.youtube.com/watch?v=tPYj3fFJGjk>

## Dropout & Regularization

 **Dropout in Neural Networks StatQuest):** [https://www.y outube.com/watch?v=ARq74QuavAo](https://www.youtube.com/watch?v=ARq74QuavAo)

## Hardware Acceleration & Vivado FPGA Implementation

**Vivado and Verilog/HDL for Neural Networks**

 **Vivado Verilog implementation of Neural Network Hands-on):** <https://www.youtube.com/watch?v=NoSPWfOh1GU>

 **FPGA Design Tutorial: Vivado & Algorithm Implementation:** <https://www.youtube.com/watch?v=jStQ3dX4Ihk>

 **Neural Network Hardware Accelerator Part-1 (blog, code, and Vivado flow):** [https://blog.devgenius.io/make-your-neural-network-hardware-accelerator-part-1 19cafdf2](https://blog.devgenius.io/make-your-neural-network-hardware-accelerator-part-1-19cafdf24904) [4904](https://blog.devgenius.io/make-your-neural-network-hardware-accelerator-part-1-19cafdf24904)

## FPGA-specific Neural Network Topologies

 **Neural Networks on FPGA YouTube Series, hardware design & coding):** Part 1: <https://www.youtube.com/watch?v=rw_JITpbh3k>

Part 2: <https://www.youtube.com/watch?v=a2wOjxRf_xg>

## Written PDF Guide

 **Full thesis, Neural Network on FPGA, Vivado, VHDL** [https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG\_Andreu\_Giménez\_v2.pdf?s](https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG_Andreu_Gim%C3%A9nez_v2.pdf?sequence=3&isAllowed=y) [equence=3&isAllowed=y](https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG_Andreu_Gim%C3%A9nez_v2.pdf?sequence=3&isAllowed=y)

## Vivado Project Tutorials – For Hands-on FPGA Implementation

### Start with a basic Verilog neural network and simulate step-by-step using Vivado simulator.

 **Reference the hardware architecture from the paper: ** Input: 10 HRV metrics (as fixed-point signals)

 Layers: 16, 32, 64 neurons in hidden layers; 1 in output layer  Use ReLU and Sigmoid activation function modules

 Use FSM (finite state machine) to coordinate layer computations  Synthesize, test, and deploy on Artix-7 (or similar) FPGA.

### Recommended tutorial for a similar FPGA-based MLP with Vivado:

 [Implementation of a Neural Network in FPGA Vivado, PDF walkthrough)](https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG_Andreu_Gim%C3%A9nez_v2.pdf?sequence=3&isAllowed=y) **Additional code examples and open-source frameworks:**

 [GitHub: fastmachinelearning/hls4ml Vivado support)](https://github.com/fastmachinelearning/hls4ml)

## Summary Table: What to Do and Why

|  |  |
| --- | --- |
| What to Do | Why It Matters/How It Improves Results |
| Use diverse HRV features | Captures more hidden signals, boosts detection accuracy |
| Tune NN layers/neurons | Optimizes learning: not too simple, not too complex |
| Dropout, regularization | Prevents model from simply memorizing (overfitting) |
| Strict validation methods | Ensures model works for new, real-world patients |
| Fixed-point FPGA model | Makes model super fast, wearable-ready, highly efficient |
| Project in Vivado | Lets you simulate, synthesize, and implement for testing |

### By following these simple, concrete steps in both model design and hardware implementation—specifically with Vivado tutorials—you can achieve highly accurate, fast, and practical heart event prediction systems, as outlined in the cited technical paper 53 .

⁂

[https://www.nature.com/articles/s41598 025 85765-x](https://www.nature.com/articles/s41598-025-85765-x)

<https://pmc.ncbi.nlm.nih.gov/articles/PMC10456609/>

[https://ccforum.biomedcentral.com/articles/10.1186/s13054 020 03103 1](https://ccforum.biomedcentral.com/articles/10.1186/s13054-020-03103-1)

<https://www.youtube.com/watch?v=i9T3AyBawUc>

<https://yce.perfectcorp.com/video-enhancer-ai>

<https://online.hitpaw.com/online-video-enhancer.html>

<https://vmake.ai/video-enhancer>

<https://www.fotor.com/video-enhancer/upscale-video-to-4k/>

<https://www.capcut.com/tools/ai-video-upscaler>

<https://flixier.com/tools/video-enhancer>

<https://www.kapwing.com/tools/video-enhancer>

<https://www.media.io/video-enhancer.html>

<https://www.lenovo.com/in/en/glossary/ultra-hd-4k-vs-hdr/>

<https://www.strong-eu.com/blog/hd-fhd-uhd-4k-what-are-the-differences>

<https://reolink.com/blog/uhd-vs-4k/>

<https://www.tcl.com/in/en/blog/guides/is-uhd-tv-4k>

<https://www.seenebula.com/blogs/how-to-select/uhd-vs-4k>

[https://support.nsdcindia.org/portal/en/kb/articles/candidate-registration-enrollment-process-on-skill-i](https://support.nsdcindia.org/portal/en/kb/articles/candidate-registration-enrollment-process-on-skill-india-digital-portal-sid) [ndia-digital-portal-sid](https://support.nsdcindia.org/portal/en/kb/articles/candidate-registration-enrollment-process-on-skill-india-digital-portal-sid)

<https://www.youtube.com/watch?v=azaxOg7FkYI>

[https://www.itigarnota.ac.in/wp-content/uploads/2024/02/User-Manual\_ITI Trainee-Non-Aadhaar-flow\_](https://www.itigarnota.ac.in/wp-content/uploads/2024/02/User-Manual_ITI-Trainee-Non-Aadhaar-flow_Profile-edit-flow.docx_S.pdf) [Profile-edit-flow.docx\_S.pdf](https://www.itigarnota.ac.in/wp-content/uploads/2024/02/User-Manual_ITI-Trainee-Non-Aadhaar-flow_Profile-edit-flow.docx_S.pdf)

<https://support.nsdcindia.org/portal/en/kb/articles/sid-ekyc-biometric-authentication>

[https://xavierscollegegoa.ac.in/wp-content/uploads/2023/12/Web-App-SID India-Skill-2023 User-Man](https://xavierscollegegoa.ac.in/wp-content/uploads/2023/12/Web-App-SID-India-Skill-2023-User-Manual.pdf) [ual.pdf](https://xavierscollegegoa.ac.in/wp-content/uploads/2023/12/Web-App-SID-India-Skill-2023-User-Manual.pdf)

<https://www.skillindiadigital.gov.in/faq>

[https://www.youtube.com/watch?v=aEItIiCS 1c](https://www.youtube.com/watch?v=aEItIiCS-1c)

<https://skillindiamission.in/bank-details>

[https://www.youtube.com/channel/UCLtBw-i1 ZgFGwcZKpM5ngw/videos](https://www.youtube.com/channel/UCLtBw-i1-ZgFGwcZKpM5ngw/videos)

<https://www.youtube.com/watch?v=azaxOg7FkYI>

[https://support.nsdcindia.org/portal/en/kb/articles/candidate-registration-enrollment-process-on-skill-i](https://support.nsdcindia.org/portal/en/kb/articles/candidate-registration-enrollment-process-on-skill-india-digital-portal-sid) [ndia-digital-portal-sid](https://support.nsdcindia.org/portal/en/kb/articles/candidate-registration-enrollment-process-on-skill-india-digital-portal-sid)

[https://www.youtube.com/watch?v=aEItIiCS 1c](https://www.youtube.com/watch?v=aEItIiCS-1c)

[https://www.youtube.com/channel/UCLtBw-i1 ZgFGwcZKpM5ngw/videos](https://www.youtube.com/channel/UCLtBw-i1-ZgFGwcZKpM5ngw/videos)

<https://www.skillindiadigital.gov.in/faq>

[https://xavierscollegegoa.ac.in/wp-content/uploads/2023/12/Web-App-SID India-Skill-2023 User-Man](https://xavierscollegegoa.ac.in/wp-content/uploads/2023/12/Web-App-SID-India-Skill-2023-User-Manual.pdf) [ual.pdf](https://xavierscollegegoa.ac.in/wp-content/uploads/2023/12/Web-App-SID-India-Skill-2023-User-Manual.pdf)

<https://www.geeksforgeeks.org/interview-experiences/qualcomm-recruitment-process/>

<https://entri.app/blog/qualcomm-recruitment-process/>

<https://www.devopsschool.com/blog/qualcomm-selection-and-interview-process-questions-answers/>

[https://www.youtube.com/watch?v=4 6amU9rFp4](https://www.youtube.com/watch?v=4x6amU9rFp4)

<https://onlinestudy4u.in/qualcomm-careers-2025/>

<https://bgibhopal.com/what-is-the-scope-of-machine-learning-for-ece-students/>

<https://www.guvi.in/blog/top-skills-to-become-a-machine-learning-engineer/>

<https://www.linkedin.com/pulse/top-in-demand-skills-eceeee-engineers-202-mr-muthuraj-m-qktmc>

[https://snu.edu.in/blogs/ece-in-the-age-of-ai-machine-learning-why -its-more-relevant-than-ever/](https://snu.edu.in/blogs/ece-in-the-age-of-ai-machine-learning-why-its-more-relevant-than-ever/)

s00542 024 05776-y.pdf

s00542 024 05776-y.pdf

<https://www.youtube.com/watch?v=NoSPWfOh1GU>

<https://www.youtube.com/watch?v=rw_JITpbh3k>

<https://www.youtube.com/watch?v=a2wOjxRf_xg>

[https://www.youtube.com/watch?v=3qtMs5jD OY](https://www.youtube.com/watch?v=3qtMs5jD-OY)

<https://www.youtube.com/watch?v=jStQ3dX4Ihk>

[https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG\_Andreu\_Giménez\_v2.pdf?sequence=3](https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG_Andreu_Gim%C3%A9nez_v2.pdf?sequence=3&isAllowed=y) [&isAllowed=y](https://upcommons.upc.edu/bitstream/handle/2117/400633/TFG_Andreu_Gim%C3%A9nez_v2.pdf?sequence=3&isAllowed=y)

[https://blog.devgenius.io/make-your-neural-network-hardware-accelerator-part-1 19cafdf24904](https://blog.devgenius.io/make-your-neural-network-hardware-accelerator-part-1-19cafdf24904)

<https://github.com/fastmachinelearning/hls4ml>

<https://resources.altium.com/p/building-neural-networks-fpgas>

s00542 024 05776-y.pdf