Received 12 January 2023; revised 9 February 2023; accepted 1 March 2023. Date of publication 15 March 2023; date of current version 29 March 2023.

*Digital Object Identifier 10.1109/OJCOMS.2023.3253789*

# MetNet: A Novel Low-Complexity Neural Network-Aided Detection for Faster-Than-Nyquist (FTN) Signaling in ISI Channels

**AMMAR ABDELSAMIE (Student Member, IEEE), IAN MARSLAND (Member, IEEE), AHMED IBRAHIM, AND HALIM YANIKOMEROGLU (Fellow, IEEE)**

Systems and Computer Engineering Department, Carleton University, Ottawa, ON K1S 5B6, Canada

CORRESPONDING AUTHOR: A. ABDELSAMIE (e-mail: ammar.sayed@carleton.ca).

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

**ABSTRACT** This paper studies the application of neural networks to Viterbi detection of FTN signals in an intersymbol interference (ISI) channel. The main contribution of this paper is to propose a receiver structure for detecting FTN signals in unknown static ISI channel. In particular, we propose a novel low-complexity neural network structure for calculating the branch metrics, and we explore its suitability for FTN signalling with channel uncertainty. We compare the proposed network, which we call the Metric Net (MetNet), to a benchmark neural network-based technique for metric calculation, the ViterbiNet, which was originally designed for ISI channels. The simulation results confirm that the MetNet outperforms the ViterbiNet, with two orders of magnitude lower complexity, and is much more resilient to channel uncertainty than the traditional Viterbi detector, which uses Euclidean distance for metric calculations. We further show that the MetNet exhibits robustness to being trained at mismatched SNR values and FTN pulse acceleration factors, meaning that the number of trained models required can be significantly reduced. Additionally, the results show that the proposed MetNet remains a favorable alternative at much higher levels of channel uncertainties. The results also reflect that we can generalize the MetNet to work with different channel models defined by different decaying factors. Finally, we show that we succeed in achieving a bandwidth efficiency gain of 33% due to FTN by using the MetNet in the presence of channel uncertainty.

**INDEX TERMS** Faster-than-Nyquist, maximum likelihood sequence estimation, AI based signal detection, spectral efficiency enhancement, intersymbol interference.

## I. INTRODUCTION

CONTEMPORARY communication systems leverage Nyquist signaling techniques to avoid intersymbol interference (ISI), however, with recent advances in silicon technology, some ISI can be handled at the receiver in order to achieve higher rates. Faster-than-Nyquist (FTN) signalling is a promising transmission technique which improves spectral efficiency within the same operating bandwidth through accelerating signal transmission by a factor of $1/\tau$, where $0 < \tau \leq 1$. This results in self-induced pulse-shaping ISI that can be handled at the receiver to a certain extent. The authors in [1] show that using uncoded root-raised cosine transmit pulses with roll-off $\beta = 0.3$, $\tau$ can go down to 0.703 without bit error rate (BER) performance loss – the minimum value of $\tau$ we can operate at without observing any performance degradation asymptotically at high SNRs, is known as the Mazo limit. This result can be achieved when using optimal detection based on maximum likelihood sequence estimation [2], or in other words, using a Viterbi detector, which is inherently optimized to work with white noise.

In this paper, we explore the performance of FTN in an ISI channel, where we have an additional and unknown source of ISI underlying within the channel, on top of the pulse-shaping ISI in FTN - this results in longer and more severe ISI. At the receiver we compare the performance of a regular Viterbi detector against neural network aided metric calculators for Viterbi detection, in the presence of channel state information uncertainty.

Different FTN receivers were explored in [3], [4]. The authors in [3] established a low complexity, sub-optimal FTN detector based on convex relaxation, primal-dual-predictor-corrector interior point method, and quantization. While the authors in [4] exploit a mathematical programming technique based on the alternating directions multiplier method to design an FTN detector for ultra-high modulation orders up to 64K with notable spectral efficiency gains. Channel estimation issues for FTN signals in ISI channels where the channel state information (CSI) is unknown were investigated [5], [6] using pilot symbols. The performance of FTN in different multi-path channels was explored in [7], [8]. Furthermore, machine learning (ML) based receivers for FTN systems were designed in [9], [10].

Other ML applications outside FTN for ISI channels without perfect CSI knowledge were explored in [11], which introduced the sliding bidirectional recurrent neural network (SBRNN), a neural network-aided receiver showing resiliency to CSI uncertainty over the model-based counterparts. This idea gave rise to other neural network (NN) aided receivers such as the BCJRNet [12], based on the BCJR algorithm [13], and ViterbiNet [14], which we will be comparing against a new proposed neural network architecture, which we call the Metric Net (MetNet).

The main contribution of this paper is to propose a receiver structure for detecting FTN signals in unknown static ISI channel. In particular, we propose a novel low-complexity neural network structure for metric computation of the Viterbi Detector, which does not require accurate channel state information during training or deployment. We present the performance of the MetNet in an FTN system under different levels of CSI uncertainty. We compare the MetNet to the ViterbiNet in terms of computational complexity and performance, as well as comparing it to the traditional method of computing metrics for Viterbi detection using Euclidean distance (ED) [15]. We also explore the MetNet's performance when trained with no noise at all, as well as tuning the number of channels used during training as well as the number of epochs. Moreover, we explore the network's ability to perform over different ISI channels governed by decaying factors, $\gamma$, instances, when its trained using a dataset generated from different channels over a range of $\gamma$-values. Further, we test the resiliency of the MetNet to being trained on different $\tau$ values than the actual. Additionally, we present the performance of the MetNet with very high levels of channel uncertainty, and finally, we report on the bandwidth efficiency gains when using the MetNet in the FTN system.

Most neural network-based receivers designed for general ISI aim to replace model-based receivers and achieve desired results through proper training and tuning, but for the most part the network-based receiver is treated as a black box [11], [16], whereas the MetNet, similar to the ViterbiNet, embeds a neural network within a known model-based detector. Moreover, some NN-based receivers in ISI channels leverage recurrent neural networks or other architectures with memory to learn the underlying correlations in ISI channels, however, the MetNet does not leverage them to keep the complexity low while still able to achieve superior BER performances.

## II. RELEVANT LITERATURE OVERVIEW

The work in this paper is motivated by previous works that investigate deep learning techniques used to train detection algorithms using simulated samples of received signals, without accurate knowledge of the underlying channel coefficients. The work in [11] proposes a technique based on recurrent neural networks (RNNs) that is trained on a diverse dataset that contains received samples from different realizations of channel conditions, which results in a detector that is robust to channel uncertainty, the detector is called a sliding-bidirectional RNN (SBRNN). The SBRNN was shown to approach the BER performance of an optimal Viterbi detector when the channel conditions are perfectly known, and to outperform the Viterbi detector under CSI uncertainty.

Following the SBRNN, another NN-based detector was developed, called the ViterbiNet, which achieved superior results to the SBRNN [14]. The ViterbiNet, just like the MetNet in this work, acts as a metric calculator which feeds likelihood values into a Viterbi detector, where conventionally the Viterbi detector would use the Euclidean distance to compute these metrics instead. Similar to the SBRNN, the ViterbiNet outperforms traditional metric calculation in Viterbi detectors in the presence of CSI uncertainty. Similar results are demonstrated when using the BCJRnet [12], which is a network that learns a factor graph representing the channel, and carries out symbol recovery using a sum-product algorithm.

Building on top of the ViterbiNet, the Meta-ViterbiNet was proposed in [20] to enable the network to stay viable with more diverse and dynamic channel conditions through rapid online retraining of the network. This approach shows advantages over the static training of a dataset generated through different channel conditions realizations for two main reasons, first being the dataset needs to be large enough to represent all the different conditions, and secondly since the second approach struggles when the channel conditions greatly deviate from training conditions. Online training represents a valuable future work for our MetNet, especially with its low complexity nature making it suitable for frequent retraining, but it is outside the scope of this work.

Other interesting deep learning-based detectors for FTN signals include the work in [9] which introduces a joint deep learning-based detector followed by a successive interference cancellation block that calculates and subtracts the interference from received signals to obtain more accurate log-likelihood ratios.

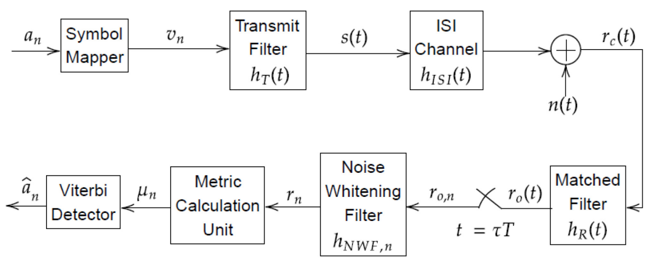


FIGURE 1. **System Block Diagram.**

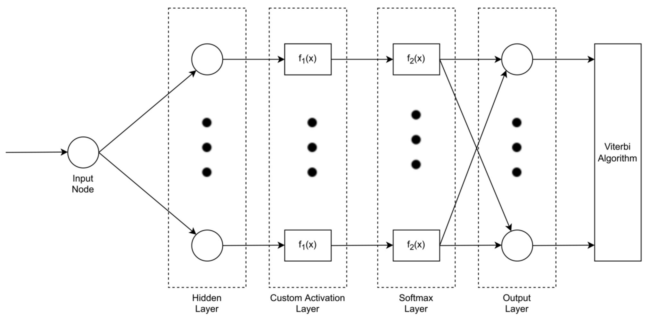


FIGURE 2. **MetNet Architecture.**

## III. SYSTEM MODEL

The system implemented is an FTN-based wireless system over an ISI channel with a Viterbi detector, as shown by the block diagram in Fig. 1. With FTN signalling, symbols are transmitted at a higher rate, but the pulse width is not changed. That is, the symbol period is reduced to $\tau T$, where $0 < \tau < 1$ is known as accelerating (or squeezing) parameter, so the transmitted signal becomes

$$s(t) = \sum_n v_n h_T(t - n\tau T), \tag{1}$$

where $\{v_n\}$ are the transmitted symbols, $T$ is the symbol period, and $h_T(t)$ is the root raised cosine pulse shape with roll-off factor $\beta$.

We consider an $L_c$-tap ISI channel, with impulse response

$$h_{ISI}(t) = \sum_{l=0}^{L_c-1} \alpha_l \delta(t - l\tau T), \tag{2}$$

where $\alpha_l$ is the gain of the $l^{th}$-tap. We also explore different levels of channel state information (CSI) at the receiver.

The combination of the root-raised cosine pulse shape, the $L_c$-tap ISI channel with channel gains of $\alpha_l$ for $l \in [0, 1, \ldots, L_c - 1]$ and additive white Gaussian noise (AWGN) with power spectral density (PSD) of $N_0$, the matched filter and the noise whitening filter, can be modeled as the equivalent discrete-time channel given by

$$r_n = \sum_{l=0}^{L-1} v_{n-l} h_l + w_n, \tag{3}$$

where $v_n$ are the transmitted symbols, $h_n$ are the effective channel taps, $L$ is the effective channel length including the ISI from both the channel and the FTN pulses, and $w_n$ is the noise, modeled by i.i.d Gaussian random variables with zero-mean and variance $N_0/2$. The effective channel taps are given by

$$h_n = \sum_{l=0}^{L_c-1} \alpha_l f_{RC,n-l}, \tag{4}$$

where $\alpha_l$ are the channel gains of the underlying ISI channel and $f_{RC,n}$ are the coefficients of the factorization of the samples of a raised cosine pulse, $h_{RC,n} = h_{RC}(n\tau T)$.

### A. METRIC COMPUTATION FOR VITERBI DETECTION

The Viterbi algorithm can be viewed as a two-step process, the first involves calculating the metrics, the next step is using those metrics to find the most likely sequence of transmitted symbols.

The branch metrics, $\mu_n(r_n|v_n, v_{n-1}, \ldots, v_{n-L+1})$, for the Viterbi algorithm at time $n$, are the negative natural log of the likelihood functions of the current received sample, $r_n$, given a hypothetical transmitted symbol, $v_n$ (corresponding to a branch between two states) and the previous $L - 1$ symbols, $v_{n-1}, \ldots, v_{n-L+1}$ (corresponding to the state). For the purposes of the VA with AWGN, this is equivalent to the more commonly used Euclidean distance (ED),

$$\mu_n(r_n|v_n, v_{n-1}, \ldots, v_{n-L+1}) = \left| r_n - \sum_{l=0}^{L-1} v_{n-l} h_l \right|^2. \tag{5}$$

This means that typically, in order to compute the metric, we require knowledge of the channel taps, $h_l$. This could be prohibitive in complex or dynamic channels which usually resort to sending reference signals often which reduces the spectral efficiency.

Alternatively, instead of computing these metrics using the Euclidean distance, we can train a neural network to learn those metrics using an offline supervised learning approach, by using simulated received signals as our the features and transmitted signals as the true labels. The MetNet, similar to the ViterbiNet, provides an alternative way of metrics computation, with the benefit of not requiring perfect knowledge of the channel when doing so. We will expand on the architectures of both the ViterbiNet as well as the MetNet in the following.

## IV. PROPOSED NEURAL NETWORK (METNET)

The motivation behind the MetNet is to train a network that aims to learn and estimate the metrics of the trellis and then uses those metrics for Viterbi detection. The aim is to also establish a significantly lower complexity alternative to the ViterbiNet, that achieves better BER performance under different levels of channel uncertainty and decaying factors in an FTN system.

The architecture of the MetNet, shown in Fig. 2 is a low complexity one, consisting of just a 1 x $M^L$ fully connected layer (FCL), followed by a custom activation layer, a softmax layer and a classification output layer. The custom activation layer applies the following function

$$f(x_i) = -|x_i|^2, \quad (6)$$

which ensures that the output of the softmax layer that follows it, closely follows a Gaussian probability density function (PDF).

The output of an $N_{out} \times N_{in}$ FCL is a vector $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{W}$ is the $N_{in} \times N_{out}$ weight matrix, and $\mathbf{b}$ is the $N_{out} \times 1$ bias vector. The weights and biases are optimized during training. The softmax layer activation function is

$$f(\mathbf{x}) = \frac{e^{x_i}}{\sum_k e^{x_k}}, \quad (7)$$

the output at the softmax layer is an estimate of the a posteriori probabilities $\Pr\{v_n, v_{n-1}, \ldots, v_{n-L+1} | r_n\}$. The last classification output layer calculates the cross-entropy loss for classification using Adam optimizer [19], which is used for training.

Prior to deployment, the neural network is trained based on a sufficient number of simulated received samples corresponding to known transmitted symbols with only imprecise knowledge of the potential state of channel during deployment. During deployment, the trained neural network is sequentially fed each received sample, $r_n$, to generate a posteriori probabilities of all $M^L$ combinations of the current and previous $L-1$ symbols, which are converted to branch metrics and fed into the Viterbi algorithm. Training for several model is only done once offline.

Whether a neural network is used for metrics computation or not, the next step remains the same within the Viterbi algorithm, the metrics calculated are used to find the shortest (survivor) path to compute an estimate for the transmitted symbols sequence.

In the next section, we perform a complexity analysis of our MetNet against the ViterbiNet.

### A. COMPLEXITY ANALYSIS AGAINST VITERBINET

The authors in [14] presented a deep neural network (DNN), called the ViterbiNet (VN). The architecture of the VN consists of a $1 \times 100$ FCL, followed by a sigmoid activation layer, a $100 \times 50$ FCL, a rectified linear unit (ReLU) activation layer, a $50 \times M^L$ FCL (where $M$ is the modulation order and $L$ is the channel length), and a softmax activation layer. The sigmoid activation function is applied to the input so that the output is bound to the interval (0,1), using the function

$$f(\mathbf{x}) = \frac{1}{1 + e^{-x_i}}. \quad (8)$$

The ReLU is simply a threshold operation that ensures the output is non-negative by setting any negative input value to zero.

Additionally, the authors incorporate a Gaussian mixture model, that is used to estimate the marginal distribution of the channel output samples, $f(r_n)$, that in turn is used to convert the a posteriori probabilities to likelihood function values according to Bayes' rule.

TABLE 1. Complexity analysis summary.

| | Complexity (flops) | For $M^L$ = 32 |
|---|---|---|
| **ViterbiNet** | 14200 +100$M^L$ | 17400 |
| **MetricNet** | $3M^L$ | 96 |
| **Euclidean Distance** | $2M^L$ | 64 |

We compare the complexity of the MetNet with the VN during the deployment phase (after the network has trained). Our analysis assumes that multiplications, additions, subtractions and divisions all count as a single floating point operation (flop), and exponentials and logarithms are implemented using $5^{th}$-order rational polynomials which translates to around 20 flops each.

The VN's first fully connected layer (FCL) of size $1 \times 100$, multiplies by the weight and adds the bias at each of the 100 nodes, which translates to 200 flops. The sigmoid activation performs an addition, a division, and an exponential at each node, translating to 2200 flops. The second $100 \times 50$ FCL requires $50 \times 100 \times 2$ flops, and the intermediate ReLU activation is a simple threshold operation, so there are no operations involved since it is just a comparison with zero. The last $50 \times M^L$ FCL requires $M^L \times 50 \times 2$ flops.

These layers are followed by the softmax activation layer, as well as additional processing to convert the a posteriori probabilities to likelihood functions and a negative log to provide the branch metrics. Although these three steps were included in the system described in [14], they serve no valuable purpose after the network has been trained, because the output of the last FCL could have been directly fed into the VA with absolutely no difference in system performance. This is because they involve converting between log and linear domains and back again, and including additive constants to all branch metrics at a given time which will not affect the VA's decisions. We therefore do not include the complexity of these three steps into our analysis. Therefore the total number of operations of the VN is $14200 + 100M^L$ flops per received sample.

Similarly, we perform the same analysis for the MetNet. The FCL we use requires $M^L \times 2$ flops, and our custom activation layer adds an extra $M^L$ flops. In total, the MetNet requires $3M^L$ flops per received sample.

For example, with $M^L = 32$, which is what we use in the simulations, the VN needs 17400 flops to calculate the branch metrics for each received sample, while the MetNet needs only 96 flops, which is two orders of magnitude better. For reference, the Euclidean distance (ED), which is just a subtraction and a square, requires $2M^L$ flops, as shown in Table 1. Therefore, the MetNet is almost as low complexity as the traditional ED used in classical VA, but achieves a much higher performance in detecting an FTN sequence under CSI uncertainty, as will be shown in the next section.

## V. SIMULATION AND RESULTS

To investigate the performance of the MetNet, Monte Carlo simulation was performed, using BPSK modulation and a 4-tap ISI channel defined by

$$\alpha_l = e^{-\gamma l}, \tag{9}$$

where $l = [0, 1, \ldots, L_c - 1]$ and $L_c = 4$. A decaying factor of $\gamma = 1$ was used for most of the results, while the squeezing parameter investigated in this study is mostly for $\tau = 0.8$, this technique would also work under the same ISI conditions without FTN, but as mentioned earlier the scope of this work is for FTN. Moreover, due to the low complexity nature of the network, it fails to perform well under severe dynamic fading, such as Rayleigh fading.

We compare the performance of a detector with known channel coefficients and another one with corrupted channel coefficients, $\tilde{h}$, such that

$$\tilde{h}_l = (1 + \epsilon_l)h_l, \tag{10}$$

where $\epsilon_l$ denotes the corruption, simulated using a zero-mean Gaussian random variable with variance $\sigma_\epsilon^2$.

During corrupted training, the neural network is trained offline on simulated received symbols generated from different instances of corrupted channel taps. In other words, the received samples, $r_n$, in (3) will be simulated using $\tilde{h}_l$ instead of $h_l$. On the other hand, the traditional ED metric calculator uses different instances of corrupted channel taps to compute the metrics, so the metrics, $\mu_n$, that are calculated based on (5) will be computed using $\tilde{h}_l$ instead of $h_l$.

### A. METRIC COMPUTATION COMPARISON

Fig. 3 shows the results established in [21] of the MetNet against the ViterbiNet (VN) and the Euclidean distance (ED) methods of metric computation at $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$, and $\gamma = 1$. This plot shows that when the receiver has noisy channel estimates, $\tilde{h}$, both of the neural network-based architectures perform significantly better than the conventional Euclidean distance method, which suffers from an error floor. As a benchmark, Fig. 3 also shows the BER for the ideal case when the ED-based metrics are used with perfect CSI. Both NN-based approaches are nearly as good despite having imperfect CSI. Further, the results show that the MetNet produces better results than the VN despite having significantly lower complexity. For example, the bit error rate (BER) of the ViterbiNet, at 8 dB, is $1.9 \times 10^{-4}$, while the MetNet has a BER of $7.5 \times 10^{-5}$, which is very close to the optimal result using the Euclidean distance with perfect CSI to calculate the metrics, which has a BER of $5.5 \times 10^{-5}$ at 8 dB. Meanwhile the ED metric with channel uncertainty showed the worst results with a BER of $2 \times 10^{-2}$ at 8 dB, which shows that it is not robust to channel uncertainty.

There was basic tuning done in [21] to generate Fig. 3, however, the next section will establish a more formal and thorough tuning of key hyperparameters.

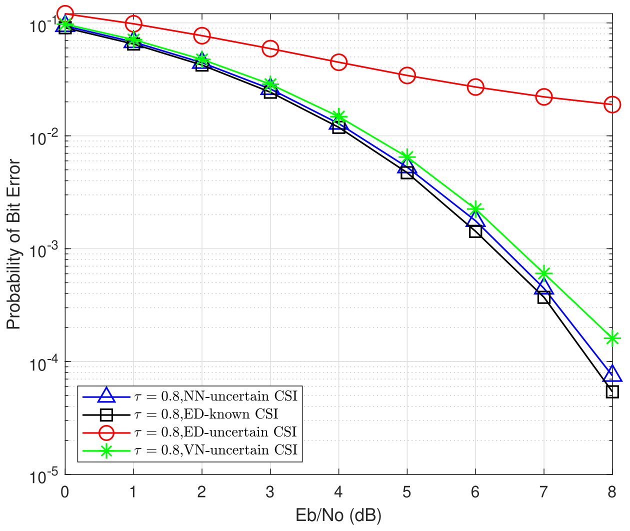


**FIGURE 3.** Probability of BER vs $E_b/N_0$ for Euclidean distance (ED), original ViterbiNet (VN), and MetNet (NN) for $\tau = 0.8$, $\gamma = 1$, and $\sigma_\epsilon^2 = 0.1$.

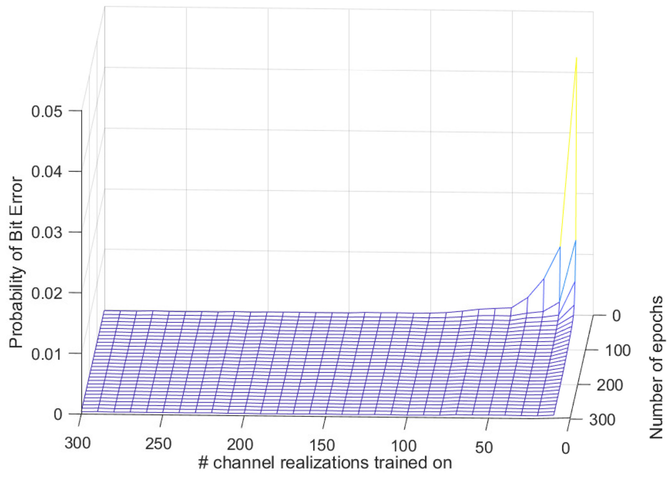


**FIGURE 4.** 3D plot of BER vs $N_{chs}$ vs $N_{eps}$.

### B. HYPERPARAMETER TUNING

Two of the most hyperparameter to tune in order to avoid a model that overfits or underfits are the training size and the number of epochs, $N_{eps}$. Since we established earlier that training benefits from having blocks of sizes $M^L$ each passed through a different channel realization, the training size is essentially determined by the number of different channels, $N_{chs}$ we train on, i.e., training size $= M^L$(number of different channels).

Hence, we tune both these parameters by comparing different models generated from different values of both parameters where we have $N_{chs} \in \{10, 20, \ldots, 300\}$ and $N_{eps} \in \{10, 20, \ldots, 300\}$. Fig. 4 shows a 3D plot showing the BER for each combination of $N_{chs}$ and $N_{eps}$, this plot is shown for models trained at $\tau = 0.8$, $\gamma = 1$ and $\sigma_\epsilon^2 = 0.1$. From this plot, we can establish that there is a trough that is lower than the rest, and we can also see that at some point, increasing the number of epochs or different channels results in diminishing returns.

TABLE 2. Hyperparameters used.

| Hyperparameter | Value |
|---|---|
| Training dataset | 5000 |
| Number of epochs | 50 |
| Loss function | Cross-entropy |
| No. of hidden units | $M^L = 32$ |
| Learning rate | 0.01 |
| Optimizing function | Adam |

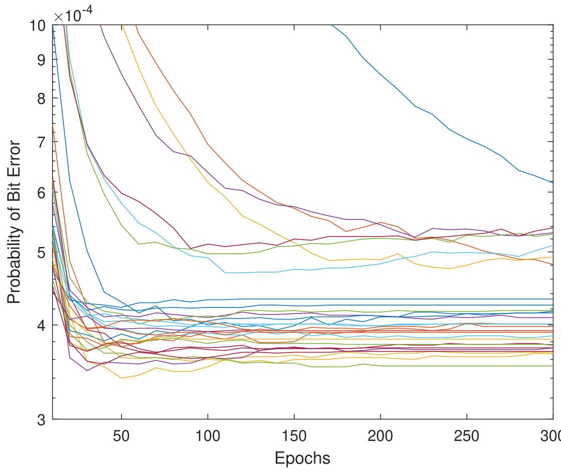
We can break this 3D plot down to two 2D plots showing the effects of $N_{chs}$ and $N_{eps}$ separately. Fig. 5(a) shows the BER vs the number of epochs, where each curve shows a different number of channel realizations trained on, and similarly Fig. 5(b) shows the BER vs the number of channel realizations trained on, where each curve shows a different number of epochs trained with. Fig. 5(a) shows that for almost all number of epochs that are high enough, training on 100 different channel realizations yields the minimum BER, similarly Fig. 5(b) achieves the minimum BER at 50 epochs. We note that the difference between the ideal (minimum BER) and other combinations of these parameters is not very significant, however, at 100 different channels with 50 epochs we found that we achieve the best results at lower complexity than other results. While one might assume that training at more channel realizations should result in better training, it could be the case that once we go higher than 100, the model starts saturating and perhaps overfits a little, which would explain the trend of slight performance dip as the number of channels trained on goes higher. The final parameters used are summarized in Table 2.

## C. TRAINING WITHOUT NOISE

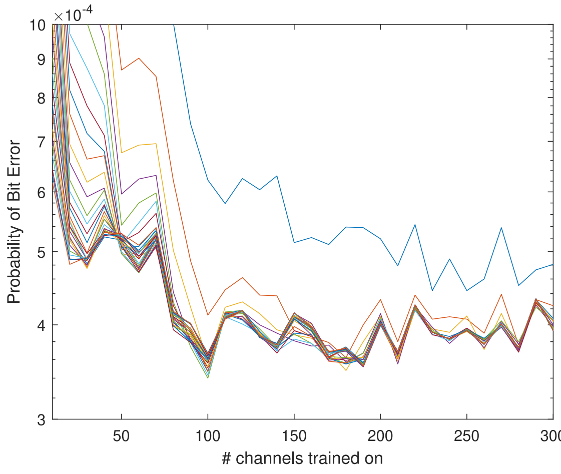
The results in Fig. 3 are generated from a system that is trained offline with a separate model at each SNR. Simulation results shows that the MetNet exhibits SNR-resiliency results that can be extrapolated to really high SNR values as shown in Fig. 6, these results show that training at very high SNR's such as 100, there is no loss in performance, essentially at such high SNR, this is equivalent to training with no noise. This approach is works because of the presence of error variance, and would not work for very low or zero error variance, as will be shown by some results shortly.

We configure the training bits to be a vector of just $M^L$ elements, corresponding to all the possible transmitted sequence bits, this block is repeated $N$ times, where $N$ is the number of different channel realizations (taps) that we decide to train on. With the previous approach, each block size was greater than $M^L$, additionally, the bits in the block were randomly generated without ensuring that each block represented all $M^L$ possible sequences.

We investigate different training approaches, including training with and without noise, as well as training with randomly generated bits blocks and with blocks of $M^L$ corresponding to all possible transmitted sequences. Fig. 7 shows the results of each approach, we found that the best approach



(a) BER vs $N_{eps}$ where each curve is for a different channel realization used.



(b) BER vs $N_{chs}$ where each curve is for a different number of epochs used.

FIGURE 5. Results showing (a) BER vs $N_{eps}$ and (b) BER vs $N_{chs}$.

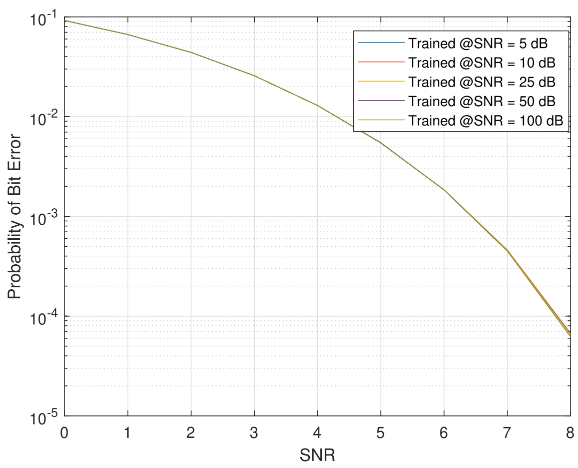


FIGURE 6. Results when training at different SNR's.

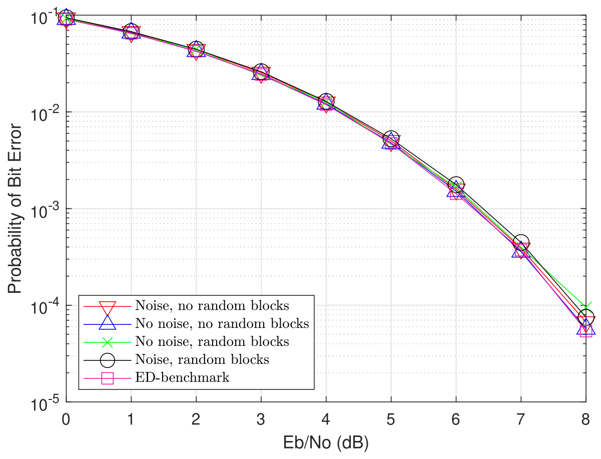


FIGURE 7. Results of different training approaches, including training with and without noise, and with specific and random bits blocks.

to use was training with no noise, with training blocks of size $M^L$, which achieves a BER $5.5 \times 10^{-4}$, very close to optimal result.

## D. PERFORMANCE AT HIGH ERROR VARIANCE

As stated earlier, training without noise works because of the presence of sufficient error variance, which adds the necessary randomness in our training dataset to generalize well even without AWGN noise, this is illustrated in this section.

The MetNet remains a better alternative for metric calculation than using Euclidean distance and the ViterbiNet, even as the error variance gets more severe, as shown by Fig. 8, which shows the BER performance of the MetNet, the ViterbiNet, and Euclidean distance over a range of different error variances. The graph demonstrates that the MetNet exhibits only a very slow performance degradation as the error variance increases, whereas the ViterbiNet degrades much more rapidly. These results are for $\tau = 0.8$, however the trend is the same regardless of $\tau$.

Another important observation is the poor performance of the MetNet at very low error variance, this is due to the fact that when trained without noise, as well as low error variance, there is very little variance in the training set, this results in overfitting and degrades the BER performance on the test set. In fact, having sufficient error variance in the first place is what allows the system to perform very well with no noise during training.

The next section dives deeper into the robustness of our NN, and tests its resilience to being trained and tested at mismatched accelerating parameters $\tau_a$ and decaying factors $\gamma$.

## E. ROBUSTNESS OF METNET

In addition to the SNR invariance demonstrated by the MetNet, as well as its ability to perform well at high error variance, the network also exhibits some $\tau$-resiliency as well.

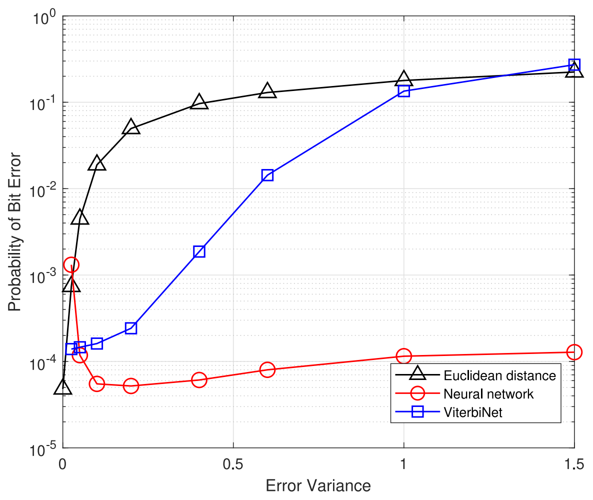


FIGURE 8. BER of different metric computation methods over different error variances for $\tau = 0.8$, and $\gamma = 1$.

TABLE 3. Performance difference when using a network trained at $\tau = 1$.

| $\tau$ tested at | SNR loss |
|---|---|
| 0.8 | 4 dB |
| 0.9 | 0.6 dB |
| 1 | 0 dB |

For example, a network trained at $\tau = 1$ still performs relatively well at $\tau = 0.9$, but that does not remain true for $\tau = 0.8$, as shown by Table 3. These results are generated at 8 dB with CSI uncertainty $\sigma_\epsilon^2 = 0.1$.

Moreover, the MetNet shows some resiliency to being trained at mismatched decaying factors, $\gamma$. Fig. 9 demonstrates these results for a network trained at $\gamma = 1$, $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR $= 8$ dB, the results suggests that when training at a lower decaying factor (more severe ISI), the MetNet can still perform relatively well on higher decaying factors (less ISI), in this particular example, for a network trained at $\gamma = 1$, the results at $\gamma = 1.2$ were still within a close BER. However, when testing at lower decay values than the one trained at, the BER drop-off is more severe and noticeable, as we can see from the Fig. 9, the result at $\gamma = 0.8$ had a much more steep BER performance drop than at $\gamma = 1.2$. Therefore, it is important to also observe the results of a network tested at $\gamma = 1$, while being trained on mismatched $\gamma$-values. Fig. 10 shows the results of a network trained at different $\gamma$-values, at $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR $= 8$ dB, and tested at $\gamma = 1$, we notice the same trend as in Fig. 9.

Building a network that has even stronger $\gamma$-resiliency is desirable, as that mean that the network is able to work better in more varied channel conditions. We build several other networks, that are trained on a dataset generated from different realizations of $\gamma$.

Next, we test the robustness of each network when it's trained on a range of $\gamma = X \pm 0.1$, and tested at $\gamma = X$. Fig. 11(a) compares the baseline results for a network trained

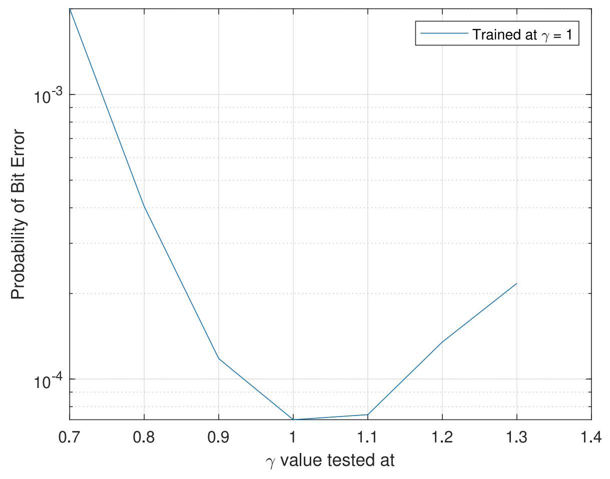


FIGURE 9. BER of different networks trained at $\gamma = 1$, $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR = 8 dB, and tested on mismatched $\gamma$ values.

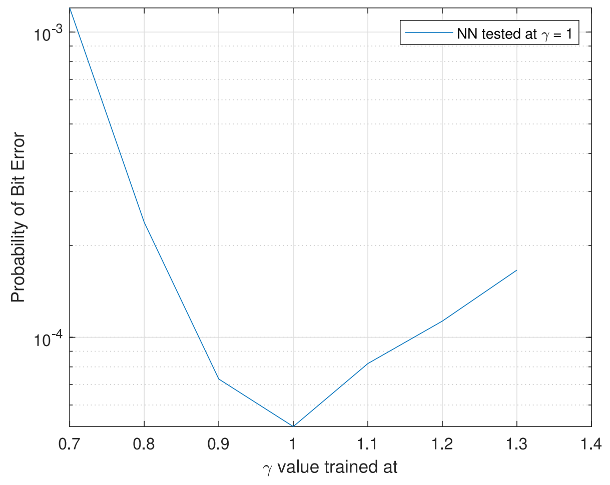
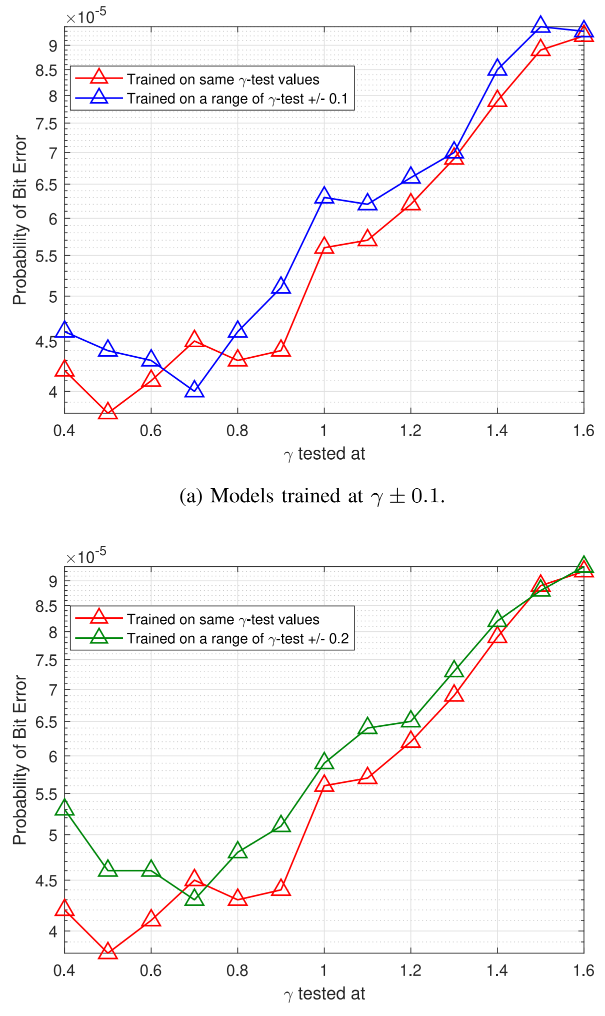


FIGURE 10. BER of different networks trained at different $\gamma$-values, $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR = 8 dB, and tested on $\gamma = 1$.



(a) Models trained at $\gamma \pm 0.1$.

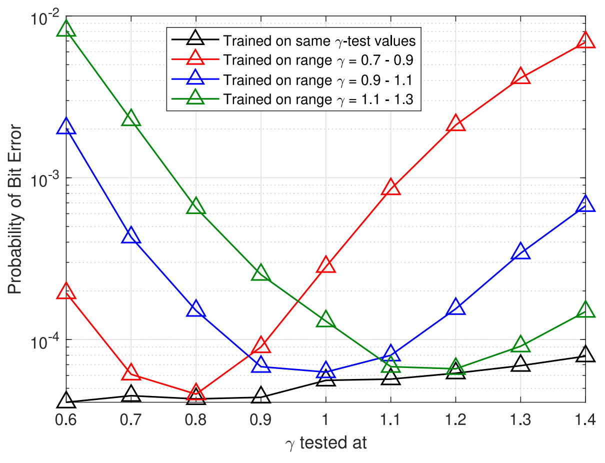
(b) Models trained at $\gamma \pm 0.2$.

FIGURE 11. Results of training a network at $\gamma = X \pm \epsilon$ values and testing at $\gamma = X$. Subfigure (a) $\epsilon = 0.1$ (b) $\epsilon = 0.2$.

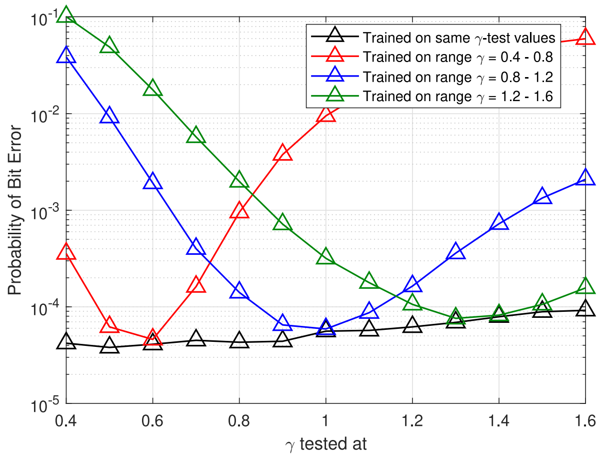
and tested at the same $\gamma$ values (red), to the results when we train on $\gamma = X \pm 0.1$, and tested at $\gamma = X$, Fig. 11(b) repeats the same experiment but the models are tested on $\gamma = X \pm 0.2$ instead, with $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR = 8 dB. These results show that we can afford training over a range without much of a BER loss, for lower $\gamma$-values, a smaller training of range of $\gamma \pm 0.1$ is preferred, however at higher $\gamma$-values we achieve good results when training over a slightly bigger range of $\gamma \pm 0.2$, meaning we can afford to train even less models at higher $\gamma$-values.

We can also test the baseline performance (which trains at every $\gamma$-value tested on), with 3 networks, trained over the following ranges $net_1 \in [0.7, 0.8, 0.9]$, $net_2 \in [0.9, 1.0, 1.1]$, and $net_3 \in [1.1, 1.2, 1.3]$, and another with the following ranges, $net_1 \in [0.4, 0.5, \ldots, 0.8]$, $net_2 \in [0.8, 0.9, \ldots, 1.2]$, and $net_3 \in [1.2, 1.3, \ldots, 1.6]$, both are tested over a sweep of a $\gamma$ range. Fig. 12 shows these results against the baseline established earlier with $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR = 8 dB. These results show we can establish an adaptive system, with way less models trained, each model valid for a certain range of decaying factors $\gamma$. We can also see again that when operating at lower $\gamma$ values, it is better to stick with a smaller range, as shown by Fig. 12(a), while at higher $\gamma$-values, we can one model that works well over a range of 5 values of $gamma \in [1.2, 1.3, \ldots, 1.5]$ as shown by Fig. 12(b), instead of training one for each realization.

The approach of training over a range can really shine in a system limited to having only a certain number of models. For instance, if we are limited to having just 5 models, we investigate the best way of training these models. Method 1 would be training at 5 discrete $\gamma$ values, and Method 2 would be training over 5 different ranges of $\gamma$. Fig. 13 shows the result of both methods, each network's result is shown within its region of operation, with $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$ and SNR = 8 dB. For Method 1, the 5 models are trained over the following values of $\gamma \in [0.4, 0.7, 1, 1.3, 1.6]$ and for Method 2, the 5 models are trained over the following uniformly distributed ranges $net_1 \in [0.2, 0.3, \ldots, 0.55]$, $net_2 \in [0.55, 0.65, \ldots, 0.85]$, $net_3 \in [0.85, 0.95, \ldots, 1.15]$, $net_4 \in [1.15, 1.25, \ldots, 1.45]$,

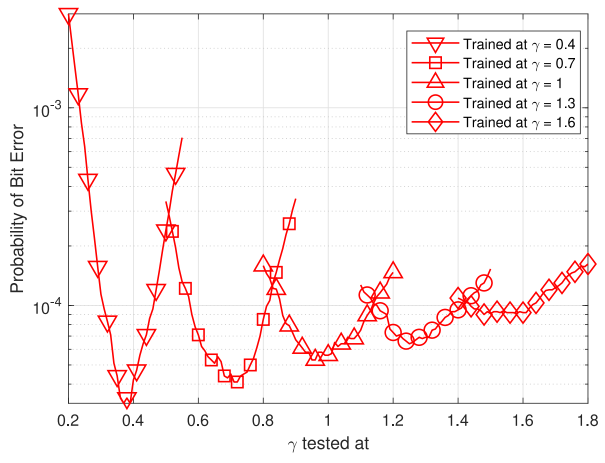


(a) Network trained on ranges of ±0.1.

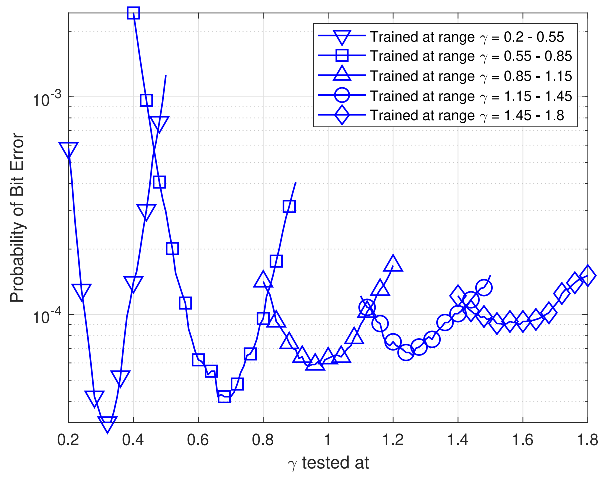


(b) Network trained on ranges of ±0.2.

**FIGURE 12.** **Results of training 3 networks at a range of ±0.1 and another of a wider range of ±0.2. Subfigure (a) range ±0.1 (b) range ±0.2.**



(a) Method 1.



(b) Method 2.

**FIGURE 13.** **Results of Method 1: Training 5 models on discrete values of $\gamma$ and Method 2: training 5 models on a range of $\gamma$-values uniformly distributed. Subfigure (a) Method 1 (b) Method 2.**

and $net_5 \in [1.45, 1.55, \ldots, 1.8]$. We do not observe a clear advantage of Method 2 over Method 1 in this setup, Method 1 is expectedly superior when tested at $\gamma$ values it was trained on, while Method 2 is marginally better at other values.

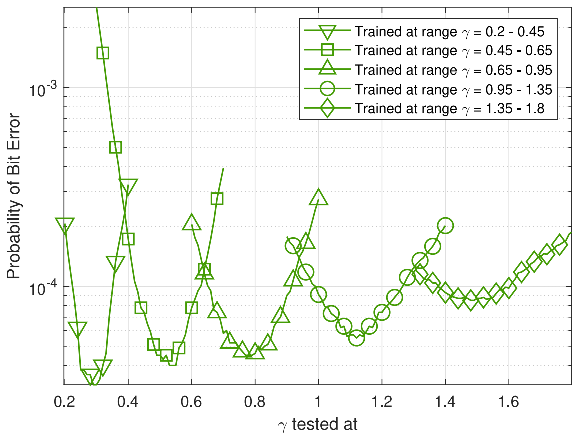
We can improve on the Method 2, based on our knowledge through previous experiments that at lower values of $\gamma$ a network performs better when its trained over smaller range of $\gamma$, hence we can have a modified version, namely Method 3, which trains over the following ranges instead $net_1 \in [0.2, 0.3, 0.45]$, $net_2 \in [0.45, 0.55, 0.65]$, $net_3 \in [0.65, 0.75, 0.85, 0.95]$, $net_4 \in [0.95, 1.05, \ldots, 1.35]$, and $net_5 \in [1.35, 1.45, \ldots, 1.8]$. Fig. 14(a) shows the results of Method 3, and Fig. 14(b) cascades all methods on the same plot as the baseline. While it might not be clear right away from Fig. 14(b) which method works best, we can compare the mean BER of each method over the testing $\gamma$ range, and the means are as follows: $\mu_1 = 1.5 \times 10^{-4}$ for Method 1, $\mu_2 = 1.08 \times 10^{-4}$ for Method 2, and $\mu_3 = 8.9 \times 10^{-5}$ for Method 3. Hence we were able to get the best results in the models-limited system using Method 3, which is training over a range of $\gamma$'s that are more narrow with lower $\gamma$ values.

Overall, we can adopt an adaptive system, which enables us to train less models with a slight losses in BER performance that is more obvious when operating at lower decaying factor $\gamma$ values where ISI is more severe.

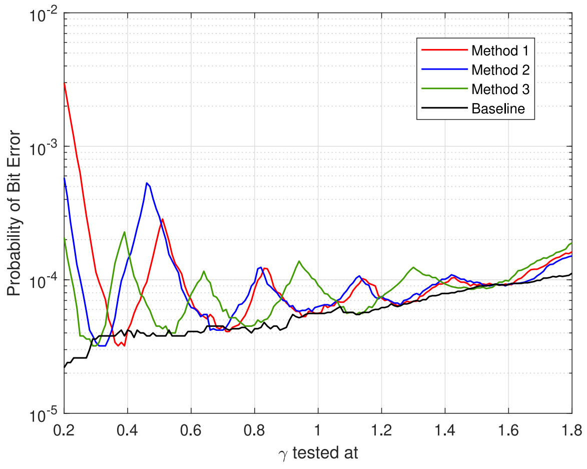
These results suggests that training time and the memory requirement can be significantly reduced – instead of training at each SNR, $\tau$ and $\gamma$ realizations, fewer networks trained with no noise and fewer $\tau$ and $\gamma$ values are sufficient and more practical.

### F. BENEFITS OF CUSTOM ACTIVATION LAYER

In the MetNet, a custom activation layer that performs $f(x) = -|x|^2$ was applies to the output of the fully connect layer. Although theoretically the inclusion of this layer may be unnecessary since the network should be able to learn without being guided by model-based knowledge, it



(a) Method 3.



(b) Method 1 (red) Method 2 (blue) Method 3 (green) and baseline (black).

**FIGURE 14. Results of Method 3: Training 5 models on a range of $\gamma$-values uniquely chosen, and the results of all methods with the baseline. Subfigure (a) Method 3 (b) All methods plus baseline.**

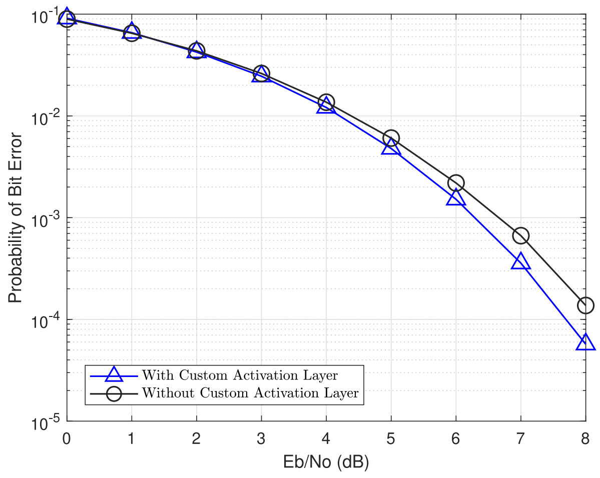


**FIGURE 15. BER results with and without using extra activation layer for $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$, and $\gamma = 1$.**

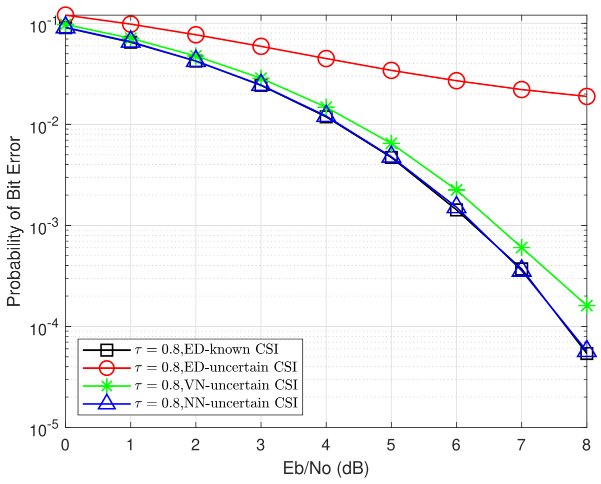


**FIGURE 16. Probability of BER vs $E_b/N_0$ for Euclidean distance (ED), original ViterbiNet (VN), and tuned MetNet (NN) that is trained with no noise for $\tau = 0.8$, $\gamma = 1$, and $\sigma_\epsilon^2 = 0.1$.**

does improve the system performance as shown by Fig. 15. The graph demonstrates that results improve when using the extra activation layer, showing an SNR gain of around 0.5 dB at a BER of $2 \times 10^{-4}$. These results are for the MetNet with CSI uncertainty at $\tau = 0.8$, $\gamma = 1$, and the same trend holds true for different individual values of $\gamma$ with varying degrees of improvement.

With that being established, we now present the results of our tuned system with the activation layer that is trained with no noise, and observe the improvements we can achieve over the results achieved earlier in Fig. 3.

## G. PERFORMANCE OF TUNED SYSTEM

In this section, we present the results of the system after the hyperparameter tuning showed earlier, with a training dataset that is simulated without AWGN, as well as with the inclusion of the activation layer. Fig. 16 shows the improved results we can achieve with our MetNet, the result we can achieve with CSI uncertainty is near-perfect, as both our NN and the optimal result using the ED have a BER of $5.5 \times 10^{-5}$ at 8 dB.

In the next section we will extend these results and test them over the average of a range of different channel models defined by different decaying factors, to show that we can maintain the performance over different channels.

## H. EFFECTS OF DIFFERENT CHANNEL MODELS

We explore the performance in 20 different channel models corresponding to 20 values of gamma and present the average BER, as was done in [14]. We average our results over a sweep of decaying factors over the range $\gamma \in \{\frac{n}{10} | n \in \{1, 2, \ldots, 20\}\}$.

Fig. 17 shows the results of the different methods of metrics computation at $\tau = 0.8$ and $\sigma_\epsilon^2 = 0.1$ values averaged over the range of $\gamma$. The results still show that the MetNet still outperforms both the ViterbiNet as well as the conventional method of metrics calculation (ED) in the presence of CSI uncertainty. For instance, at 8 dB, we see that the

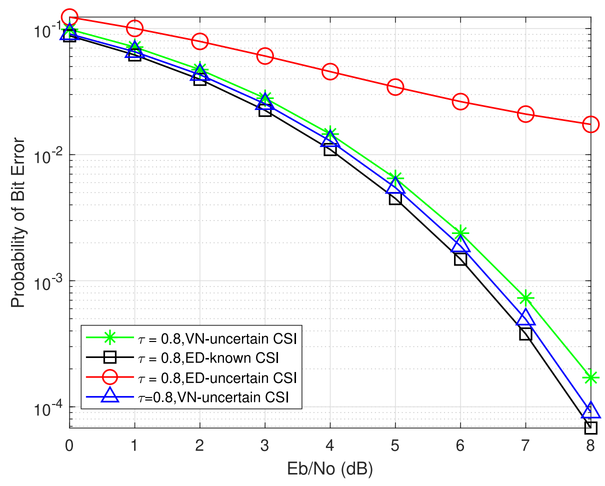


**FIGURE 17.** Comparison of different metric computations methods for $\tau = 0.8$, $\sigma_\epsilon^2 = 0.1$, averaged over different $\gamma$ values.

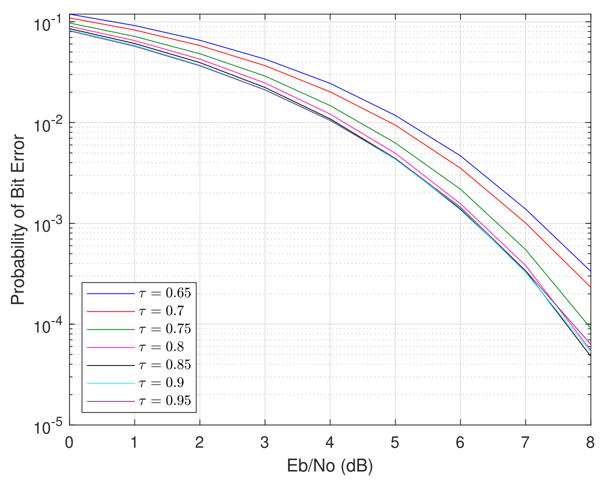


**FIGURE 18.** Results using the MetNet for different $\tau$ values, with $\sigma_\epsilon^2 = 0.1$, and $\gamma = 1$.

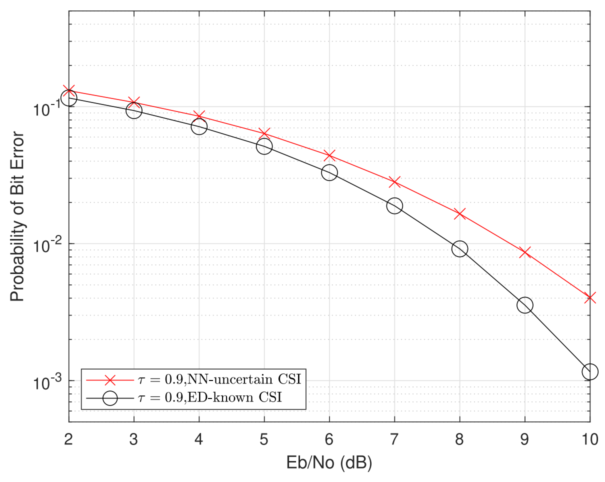


**FIGURE 19.** Result of MetNet against traditional VA at 16QAM.

optimal BER using a VA with perfect CSI knowledge is $7 \times 10^{-5}$, while the MetNet, with channel corruption has a BER $8 \times 10^{-4}$ and the ViterbiNet has a BER of $2 \times 10^{-4}$. Finally, the traditional metric computation method using ED has a much higher BER of $2 \times 10^{-2}$, seemingly approaching an error floor.

### I. SPECTRAL EFFICIENCY GAINS

Another interesting result is showing the possible bandwidth gains specific to the proposed neural network-aided detection in the ISI FTN system. Fig. 18 shows the BER curves when using the MetNet architecture, with CSI uncertainty at different values of $\tau$. There is a slight BER performance degradation at $\tau = 0.75$, meaning that this is where the Mazo limit of the system roughly lies. Hence, the bandwidth efficiency improvement for the system using the MetNet under channel uncertainty is around 33%. By comparison, when the CSI is perfectly known and the ED metric is used, it is possible to get a bandwidth efficiency gain of 39%, which is only slightly higher than for the imperfect CSI case.

### J. HIGHER ORDER MODULATION

The previous simulations were done using BPSK constellation, the network achieves identical results to Fig. 16 when tested at 4-QAM.

For 16-QAM, the results in Fig. 19 show that the MetNet is still able to recover the transmitted data, albeit with a slight performance penalty. This penalty is perhaps due to the fact that after ISI, the distribution of the received samples is too dense, so the MetNet is unable to always distinguish between different transmitted symbols.

## VI. CONCLUSION

In this paper we explored the performance of an FTN system in an ISI channel, using Viterbi detection with different methods of metric computation. We proposed a novel low complexity neural network architecture to calculate metrics, the MetNet, and compare it to another neural network-based metric calculator, the ViterbiNet, as well as the traditional method of calculating metrics using the Euclidean distance. We present the results of all methods when there is CSI uncertainty present in the system.

The results show that under uncertain channel conditions, both neural network approaches show significantly better results than the Euclidean distance-based approach. Additionally, we show through a complexity analysis that the MetNet has very low complexity that approaches the simplicity of Euclidean distance, as well as better performance compared to the ViterbiNet.

We establish that our MetNet performs better when trained without noise, and we tune the network to set the number of channel instances we train our model with as well as the suitable number of epochs. We further explore the resiliency of the MetNet by training it over a range of decaying factors, $\gamma$, instead of training a different model at each instance, as well as the resiliency to being trained at mismatched accelerating parameters $\tau$ values, these results show that we are able to significantly reduce the number of models trained, and as such we establish an adaptive system that can adopt suitable models based on the channel conditions. The results further show that the MetNet remains the favorable alternative for much more intense CSI uncertainties, where it showed strong robustness and only a slow degradation of BER performance as the error variance increased.

We showed that these results can be generalized for different channel models with different decaying factors, as the trends of the result held true when averaged over a sweep of different $\gamma$ values. Moreover, we present results that reflect a BER performance enhancement when using our custom activation layer, which makes the output of the network closely represent a Gaussian PDF. Finally, we present the bandwidth efficiency gains due to FTN when using the MetNet, with CSI uncertainty over different decaying factors, the results show that we can have a bandwidth efficiency improvement of around 33%.

There are several directions for potential future work regarding this study. To begin with, this neural network is built on top of the VA, meaning that the it still has the same $M^L$ complexity when it comes to the modulation order and the channel length, this could prove to be limiting for a higher order modulation systems. Exploring the potential of this neural network on top of a reduced complexity VA, either through channel shortening [18], reducing the trellis size [17], or other schemes to reduce the $M^L$ complexity is an interesting direction. This would enable higher order modulation, where the complex part of the signal could be handled by an identical NN structure, one part for the real component of the signal and another for the complex part, the low-complexity nature of the MetNet would work well with this approach.

Moreover, exploring the application of this NN in a multi-carrier FTN system [1], as well as channel coding [9], would allow us to reach much higher spectral efficiency gains. Additionally, due to the low complexity nature of the MetNet, it is not suitable for deep dynamic fading, such as Rayleigh fading. Exploring model enhancements that can handle Rayleigh fading is an interesting direction and is left for future work.

Finally, due to its similar nature to the ViterbiNet, the MetNet has the potential to be utilized in an online-training fashion, similar to the Meta ViterbiNet [20], where the model is able to update in real time based on incoming signals, resulting in a more resilient and dynamic model.

## REFERENCES

[1] J. B. Anderson, F. Rusek, and V. Öwall, "Faster-than-Nyquist signaling," *Proc. IEEE*, vol. 101, no. 8, pp. 1817–1830, Aug. 2013.

[2] R. Chang and J. Hancock, "On receiver structures for channels having memory," *IEEE Trans. Inf. Theory*, vol. 12, no. 4, pp. 463–468, Oct. 1966.

[3] A. Ibrahim, E. Bedeer, and H. Yanikomeroglu, "A novel low complexity faster-than-Nyquist signaling detector based on the primal-dual predictor-corrector interior point method," *IEEE Commun. Lett.*, vol. 25, no. 7, pp. 2370–2374, Jul. 2021.

[4] A. Ibrahim, E. Bedeer, and H. Yanikomeroglu, "A novel low complexity faster-than-Nyquist (FTN) signaling detector for ultra high-order QAM," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 2566–2580, 2021.

[5] T. Hirano, Y. Kakishima, and M. Sawahashi, "TDM based reference signal multiplexing for faster-than-Nyquist signaling using OFDM/OQAM," in *Proc. IEEE Int. Conf. Commun. Syst.*, 2014, pp. 437–441.

[6] N. Wu, W. Yuan, Q. Guo, and J. Kuang, "A hybrid BP-EP-VMP approach to joint channel estimation and decoding for FTN signaling over frequency selective fading channels," *IEEE Access*, vol. 5, pp. 6849–6858, 2017.

[7] Y. Yamada, M. Sawahashi, and K. Saito, "Performance of time and frequency compression of faster-than-Nyquist signaling in frequency-selective fading channels," in *Proc. 21st Asia–Pacific Conf. Commun. (APCC)*, 2015, pp. 550–554.

[8] J. T. Wang, "Performance analysis for faster-than-Nyquist signaling under multi-path channel," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 302–306, Feb. 2020.

[9] P. Song, F. Gong, Q. Li, G. Li, and H. Ding, "Receiver design for faster-than-Nyquist signaling: Deep-learning-based architectures," *IEEE Access*, vol. 8, pp. 68866–68873, 2020.

[10] B. Liu, S. Li, Y. Xie, and J. Yuan, "A novel sum-product detection algorithm for faster-than-Nyquist signaling: A deep learning approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5975–5987, Sep. 2021.

[11] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5663–5678, Nov. 2018.

[12] N. Farsad, N. Shlezinger, A. J. Goldsmith, and Y. C. Eldar, "Data-driven symbol detection via model-based machine learning," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, 2021, pp. 571–575.

[13] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.

[14] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3319–3331, May 2020.

[15] G. D. Forney Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[16] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *Proc. Conf. Inf. Sci. Syst. (CISS)*, 2017, pp. 1–6.

[17] A. Prlja and J. B. Anderson, "Reduced-complexity receivers for strongly narrowband intersymbol interference introduced by faster-than-Nyquist signaling," *IEEE Trans. Commun.*, vol. 60, no. 9, pp. 2591–2601, Sep. 2012.

[18] J. Fan, Y. Ren, Y. Zhang, and X. Luo, "MLSE equalizer with channel shortening for faster-than-Nyquist signaling," *IEEE Photon. Technol. Lett.*, vol. 30, no. 9, pp. 793–796, May 1, 2018.

[19] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980.*

[20] T. Raviv, S. Park, N. Shlezinger, O. Simeone, Y. C. Eldar, and J. Kang, "Meta-ViterbiNet: Online meta-learned Viterbi equalization for non-stationary channels," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2021, pp. 1–6.

[21] A. Abdelsamie, I. Marsland, A. Ibrahim, and H. Yanikomeroglu, "Novel low-complexity neural network aided detection for FTN signalling in ISI channel," in *Proc. IEEE Global Commun. (Globecom) Conf.*, Dec. 2022, pp. 825–830.