國立清華大學 電機工程學系

實作專題研究成果摘要

A Proposed Neural Min-sum Decoder for 5G Communication Technology

基於神經網絡的最小和解碼器於第五代行動通訊解碼之應用

專題領域：通訊領域

組　　別：B446

指導教授：翁詠祿 教授

組員姓名：顏誠佑、黃伊宏、吳明叡

研究期間：2024年2月 至 2024年11月止，共10個月

摘要

隨著通訊技術的進步，錯誤更正碼在確保數據正確性方面變得愈加重要。為了兼顧傳輸速度，現今的第五代行動通訊技術(5th Generation Communication Technology, 5G)主要改為使用較短碼長的奇偶校驗矩陣(parity check matrix)，如 BCH 碼(Bose-Chaudhuri-Hocquenghem code)來進行編解碼。然而，作為主流解碼器之一的最小和解碼器(min-sum decoder)，在處理這些短碼長奇偶校驗矩陣時，表現並不理想。針對這一問題，近年來許多學者提出利用機器學習(machine learning)來改進該解碼器的演算法[1]、[2]、[3]，並取得顯著的性能提升。

在本研究中，我們主要探討如何透過引入神經網路來提升最小和解碼器的效能，並嘗試進一步優化現有文獻中的神經網路架構[4]。除了加入額外的可訓練參數外，我們也引入了知識蒸餾(knowledge distillation)的概念，設計了新的損失函數(loss function)來優化參數學習。在使用不同碼長和碼率的 BCH 碼進行解碼錯誤率測試後，我們發現優化過後的解碼器在運算複雜度上僅為傳統模型的六分之一。

目錄

1. 研究背景

超可靠低延遲通信(Ultra-Reliable Low-Latency Communications, uRLLC)是現今5G網絡的關鍵特性之一，廣泛應用於對延遲和可靠性要求極高的工業自動化、物聯網等領域。在錯誤更正碼的編碼部分，為滿足 uRLLC 的低延遲需求，通常需要採用較短的碼長進行編解碼，以降低編解碼器的複雜度，進而提升編解碼效率。然而，這也導致解碼錯誤率的上升，進而削弱通訊的可靠性。因此，亟需更低錯誤率的短長度解碼器，來滿足 5G 網絡對可靠度的需求。

2. 研究目的

置信度傳播解碼器(belief propagation decoder)是現今常使用的解碼演算法之一，以下將簡單介紹此演算法。

首先對於一任意二進位編碼的奇偶校驗矩陣，我們均可以繪製其坦納圖(Tanner graph)，該圖說明了此奇偶校驗矩陣的檢查節點(check node, CNs)與變數節點(variable node, VNs)的連接關係，對於第 c 個檢查節點，其所連接的變數節點集合為$\mathcal{N}(\mathrm{c})$。同理，第 v 個變數節點所連接的檢查節點集合為$\mathcal{N}(\mathrm{v})$。解碼器將以此圖為基礎構架，將數據在檢查節點及變數節點之間反覆傳遞運算。

對於一個碼長為 n 的二進位已編碼碼字$\mathrm{c}^{\mathrm{n}} = (\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3, \dots, \mathrm{c}_{\mathrm{n}}), \mathrm{c}_{\mathrm{i}} \in \{0,1\}, \forall\, \mathrm{i}$，進行二位元相位移調變(binary phase-shift keying, BPSK)後得到碼字$\mathrm{x}^{\mathrm{n}} = \{\pm 1\}^{\mathrm{n}}$並由接收端送出。在經過可加性高斯白雜訊通道(additive white Gaussian noise, AWGN)後於接收端收到碼字$\mathrm{z}^{\mathrm{n}} = \mathrm{x}^{\mathrm{n}} + \mathrm{y}^{\mathrm{n}}$，其中$\mathrm{y}^{\mathrm{n}} \sim N(0, \sigma^2)$。

當解碼器收到通道傳來的碼字z^{n}，會先將其轉換成對數似然率(log-likelihood ratio)形式後，將逐位元形式送入對應的變數節點：

$$\mu_{\mathrm{ch},\mathrm{v}_i} = \mathrm{LLR}(\mathrm{z}_{\mathrm{j}}) = \log\left(\frac{\mathrm{P_r}(\mathrm{c}_{\mathrm{j}} = 0 | \mathrm{y})}{\mathrm{P_r}(\mathrm{c}_{\mathrm{j}} = 1 | \mathrm{y})}\right) = \frac{2\mathrm{z}_{\mathrm{j}}}{\sigma^2} \tag{1}$$

接下來變數節點會傳遞以上資訊至其所連接的檢查節點，於檢查節點進行以下計算後返還回各變數節點。(其中$\mu_{\mathrm{c}_j \to \mathrm{v}_i}$表示第 j 個檢查節點傳遞至第 i 個變數節點的數據；$\mu_{\mathrm{v}_i \to \mathrm{c}_j}$ 表示反向傳遞的數據；ℓ 則表示迭代的次數)

$$\mu^{(\ell)}_{\mathrm{c}_j \to \mathrm{v}_i} = 2\tanh^{-1}\left[\prod_{\mathrm{v} \in \mathcal{N}(\mathrm{c}_j) \setminus \mathrm{v}_i} \tanh\left(\frac{1}{2} \cdot \mu^{(\ell)}_{\mathrm{v} \to \mathrm{c}_j}\right)\right] \tag{2}$$

變數節點收到資訊後進行以下運算，將結果傳遞回檢查節點，進行重複迭代運算。

$$\mu^{(\ell)}_{\mathrm{v}_i \to \mathrm{c}_j} = \mu_{\mathrm{ch},\mathrm{v}_i} + \sum_{\mathrm{c} \in \mathcal{N}(\mathrm{v}_i) \setminus \mathrm{c}_j} \mu^{(\ell-1)}_{\mathrm{c} \to \mathrm{v}_i} \tag{3}$$

重複執行(2)、(3)指定迭代次數ℓ後，於變數節點作數據統整，也就是根據下式計算後驗(a posteriori)對數似然比(log-likelihood ratio, LLR)為:

$$\mu_{\mathrm{v}_i}^{(\ell)} = \mu_{\mathrm{ch},\mathrm{v}_i} + \sum_{\mathrm{c}\in\mathcal{N}(\mathrm{v}_i)} \mu_{\mathrm{c}\to\mathrm{v}_i}^{(\ell)} \tag{4}$$

最後將以上結果轉換為硬判決(hard decision)輸出，得到解碼後的碼字$\hat{\mathrm{v}}^{\mathrm{n}}$：

$$\hat{\mathrm{v}}_{\mathrm{j}} = \begin{cases} 1, & \text{if } \mu_{\mathrm{v}_i}^{(\ell)} < 0 \\ 0, & \text{else} \end{cases} \tag{5}$$

由於以上檢查節點的計算牽涉到雙曲正切函數(hyperbolic tangent)的運算，大幅提高了此演算法的運算複雜度，因此最小和解碼器(min-sum decoder)的算法被提出[5]，即置信度傳播解碼器的簡化版本，透過估計的方式將上述演算法中的(3)式改為(6)式，雖然正確率略為下降，但可有效降低運算複雜度及解碼時間。

$$\mu_{\mathrm{c}_j\to\mathrm{v}_i} = \min_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i} \left|\mu_{\mathrm{v}\to\mathrm{c}_j}\right| \cdot \prod_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i} \operatorname{sign}\left(\mu_{\mathrm{v}\to\mathrm{c}_j}\right) \tag{6}$$

有鑑於最小和的近似方法往往會高估資訊的數值大小，因此另有兩種資訊修正方案[6]，透過新增補償常數α或β，將(6)式再分別修改為

$$\mu_{\mathrm{c}_j\to\mathrm{v}_i} = \alpha \cdot \min_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i} \left|\mu_{\mathrm{v}\to\mathrm{c}_j}\right| \cdot \prod_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i} \operatorname{sign}\left(\mu_{\mathrm{v}\to\mathrm{c}_j}\right) \tag{7}$$

和

$$\mu_{\mathrm{c}_j\to\mathrm{v}_i} = \max\left(\min_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i} \left|\mu_{\mathrm{v}\to\mathrm{c}_j}\right| - \beta, 0\right) \cdot \prod_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i} \operatorname{sign}\left(\mu_{\mathrm{v}\to\mathrm{c}_j}\right) \tag{8}$$

使之從原本的最小和解碼轉變為正規化最小和(normalized min-sum, NMS)解碼或是補償式最小和(offset min-sum, OMS)解碼。

以上的解碼器演算法，在低密度奇偶校驗碼(LDPC code)中具有極佳的表現，不過 LDPC 碼的平均碼長通常較長，並不適用於現今的 uRLLC 通訊，而對於碼長較短的高密度奇偶校驗碼(HDPC code，如 BCH 碼等)，該演算法的解碼表現將大幅下滑，這是因為 HDPC 碼在坦納圖架構中有大量的小迴圈結構(4-cycle)，這使得上述演算法的迭代過程容易出現封閉式的資訊傳輸，進而加大解碼出錯的機率。

觀察坦納圖及上述演算法運作的流程，我們發現到若將整個迭代的過程展開，其實會與深度學習(deep learning)的神經網路結構高度相似。本研究以此點作出發，嘗試將上述演算法改良成可訓練的神經網路模型，期許能進一步提升該演算法在低碼長的 BCH 碼的解碼表現。

3. 研究方法

結合多篇文獻的設計理念，我們嘗試建立一個最小和算法(min-sum)的神經網絡模型，並調制相關參數使其有較佳的解碼正確率。

3.1 基礎神經網路架構

參考[1]的設計理念，我們首先將原本的補償式最小和演算法作展開(unfolded)，每一次變數節點或檢查節點的計算都視為一層隱藏層結構，因此一個ℓ次迭代的最小和解碼器就可以展開成一層輸入層(對數似然率的計算)、2ℓ層隱藏層及一層輸出層。

為了方便測試解碼器的性能，我們需要架設一個完整的通訊系統來模擬。首先在發射端亂數產生一組 k 位元的資訊位元(information bit)，與指定編碼的生成矩陣(generator matrix)相乘以後得到碼字(codeword)，經過 BPSK 調變以後送入通道，加上一高斯雜訊後進入接收端的解碼器。

在解碼器的輸入層，先計算好對數似然率以供後續計算：

$$\mu_{\mathrm{ch,v}_i} = \mathrm{LLR}(\mathrm{z_j}) = \log\left(\frac{\mathrm{P_r}(\mathrm{c_j}=0|\mathrm{y})}{\mathrm{P_r}(\mathrm{c_j}=1|\mathrm{y})}\right) = \frac{2\mathrm{z_j}}{\sigma^2} \tag{9}$$

在隱藏層中，我們設定奇數層做變數節點的計算、偶數層做檢查節點計算。對於第 t 次迭代，其對應隱藏層的變數節點計算公式如下：

$$\mu_{\mathrm{v}_i\to\mathrm{c}_j}^{(\ell)} = \mu_{\mathrm{ch,v}_i} + \sum_{\mathrm{c}\in\mathcal{N}(\mathrm{v}_i)\setminus \mathrm{c}_j} \mu_{\mathrm{c}\to\mathrm{v}_i}^{(\ell-1)} \tag{10}$$

其對應隱藏層的檢查節點計算公式如下:

$$\mu_{\mathrm{c}_j\to\mathrm{v}_i}^{(\ell)} = \max\left(\min_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i}\left|\mu_{\mathrm{v}\to\mathrm{c}_j}^{(\ell)}\right| - \beta_{\mathrm{c}_j\to\mathrm{v}_i}^{(\ell)}, 0\right)\cdot\prod_{\mathrm{v}\in\mathcal{N}(\mathrm{c}_j)\setminus\mathrm{v}_i}\mathrm{sign}\left(\mu_{\mathrm{v}\to\mathrm{c}_j}^{(\ell)}\right) \tag{11}$$

對於原本的補償參數β，我們將其設定成可訓練參數，同時拓展其維度，讓坦納圖中的每一條邊(edge)都有一個對應的參數，讓每一條邊的資訊權重不再相等，其目的在於更精確地控制節點之間的資訊傳遞，透過訓練的方式來抓出較重要節點的資訊量值，以增加解碼準確度。

在最後的輸出層，會先針對各個變數節點計算後驗對數似然比:

$$s_v^{(\ell)} = \mu_{\mathrm{ch,v}_i} + \sum_{\mathrm{c}\in\mathcal{N}(\mathrm{v}_i)} \mu_{\mathrm{c}\to\mathrm{v}_i}^{(\ell)} \tag{12}$$

當訓練模型(training)時，輸出層將根據以上資訊計算各位元的交叉熵損失函數(binary cross entropy loss)，並以此進行梯度下降(gradient descent)來對參數進行更新。其公式如下式。

$$\mathrm{L_{ce}} = \frac{-1}{\mathrm{N}} \sum_{\mathrm{v}=1}^{\mathrm{k}} \mathrm{B_v} \log(\sigma(s_v)) + (1 - \mathrm{B_v}) \log(1 - \sigma(s_v)) \tag{13}$$

此處的$\sigma(\cdot)$函式為 sigmoid 激活函數，其公式為 $\sigma(x) = 1/(1 + \exp(-\mathrm{x}))$；$\mathrm{B_v}$則為傳送端未加上雜訊的傳輸資料，$\mathrm{B_v} \in \{-1, +1\}$，即正確答案。

在測試模型(testing)時，則直接以s_v進行硬判決輸出，公式同前述(5)式，再與接收端的資訊位元作比對，以此計算位元錯誤率(bit error rate, BER)。

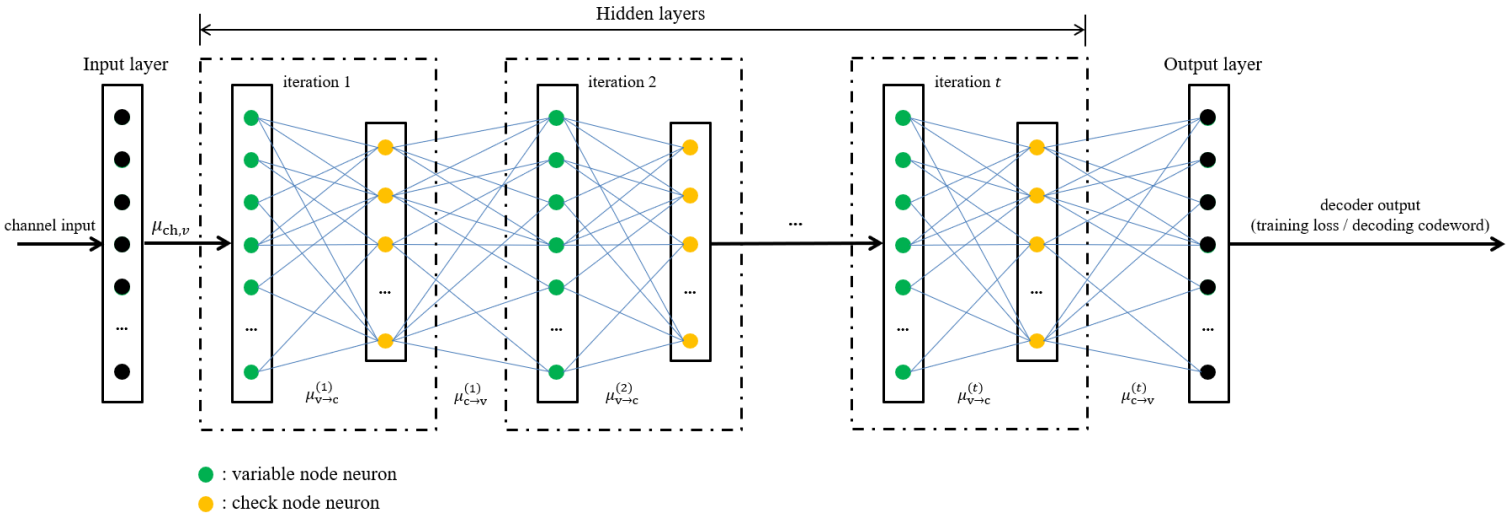


圖 1 最小和算法的基礎神經網路架構圖

以上的神經網路解碼器模型，根據[3]的命名方式，我們將其命名為神經補償式最小和解碼器(Neuron Offset Min Sum decoder, NOMS)，而接下來將針對此基本模型進行改良，嘗試使其有更好的表現。

3.2 知識蒸餾法架構及損失函數設計

知識蒸餾是機器學習中常見的模型壓縮技術，在知識蒸餾的架構底下通常會有兩個模型，分別是教師模型(teacher model)和學生模型(student model)。教師模型是一個龐大且複雜的神經網路，雖然擁有較高的準確率，卻會消耗不少運算資源；相較之下，學生模型的複雜度較低，但是無法提供媲美教師模型的預測結果。而知識蒸餾的方法旨在萃取出教師模型的精華給學生模型使用，讓低複雜度的學生模型也能達到跟複雜的教師模型相近的性能。

我們參考[7]的作法，將做 30 次迭代的最小和解碼器作為教師模型($T_{teacher} = 30$)，而學生模型則是使用做 5 次迭代的神經最小和解碼器($T_{student} = 5$)。由於在置信度傳播的演算法中，較高的迭代次數通常能夠提高推理結果的準確性，因此我們將教師模型中最後 5 次迭代的訊息視為精華，並且把教師模型和學生模型中檢查節點上訊息的差異程度當成訓練學生模型時的損失(loss)大小。損失函數可寫成下列數學式

$$L_{kd} = \sum_{\ell=1}^{T_{student}} \sum_{v=1}^{n} \sum_{c \in \mathcal{N}(v_i)} \left\| \sigma\left(\mu^{(\ell+25)}_{c_j \to v_i, teacher}\right) - \sigma\left(\mu^{(\ell)}_{c_j \to v_i, student}\right) \right\|_p^p \qquad (14)$$

其中 $\sigma(\cdot)$為 sigmoid 激勵函數(activation function)、p 為範數級數(norm order)。

在訓練的過程中，模型會根據計算出的梯度不斷地更新參數，讓學生模型中檢查節點上的訊息和教師模型對應的訊息越來越相近，進而達到降低損失的目標。因此，藉由模型參數的訓練以及使用的損失函數，教師模型得以引導學生模型去模仿其經過 25 次迭代運算後檢查節點上的精華訊息，使學生模型的性能有所提升卻依然保有低複雜度的優勢。

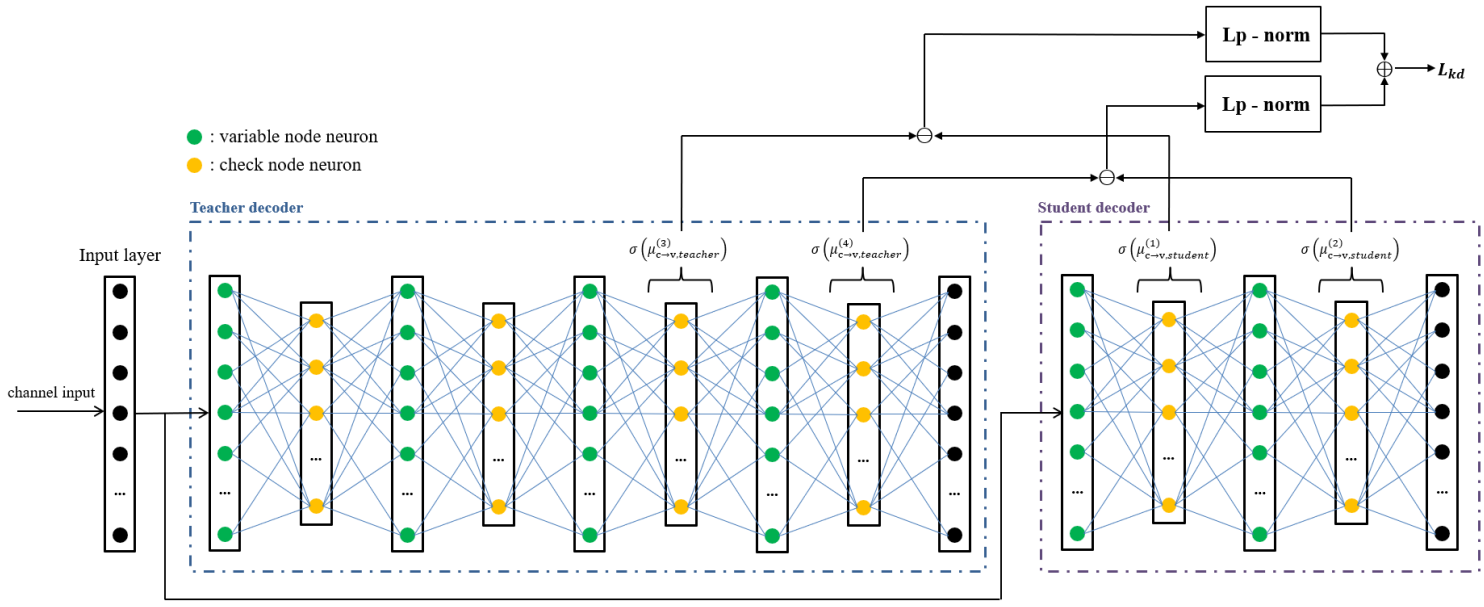


圖 2 用知識蒸餾技術訓練模型的示意圖，其中 $T_{teacher} = 4$，$T_{student} = 2$

3.3 修改版知識蒸餾法(modified knowledge distillation, mKD)

為了讓學生模型能夠更貼近甚至超越教師模型的性能，我們提出了修改版的知識蒸餾架構。

由於我們使用的是 BCH 碼，其奇偶校正矩陣對應的坦納圖中存在許多短迴圈(short cycle)，這些短迴圈有利於錯誤傳播，使得置信度傳播演算法的估計結果變得不可靠，影響解碼的性能。此外，我們在訓練神經補償式最小和(neural offset min-sum, NOMS)解碼器的過程中觀察到模型中唯一的可訓練參數(即(11)式中的$\beta_{\mathrm{c}_j \to \mathrm{v}_i}$)會隨著訓練的進行而變得越來越大，使得檢查節點上的訊息數值變小甚至為 0，這等同於將坦納圖中的某些邊切斷。因此可以合理推論神經補償式最小和解碼器在解碼性能上優於最小和解碼器的貢獻來源部分源自於坦納圖中短迴圈數量的減少。

根據上述觀點，我們將「降低坦納圖中短迴圈數量」的想法納入知識蒸餾的架構中。我們首先在每條從檢查節點連接到變數節點的邊上加上權重，並將初始值設為 1，因此檢查節點到變數節點的訊息傳遞將遵循以下的數學式

$$\mu_{\mathrm{c}_j \to \mathrm{v}_i}^{(\ell)} = w_{\mathrm{c}_j \to \mathrm{v}_i}^{(\ell)} \cdot \max\left(\min_{\mathrm{v} \in \mathcal{N}(\mathrm{c}_j) \setminus \mathrm{v}_i} \left(\left| \mu_{\mathrm{v} \to \mathrm{c}_j}^{(\ell)} \right| \right) - \beta_{\mathrm{c}_j \to \mathrm{v}_i}^{(\ell)}, 0 \right) \cdot \prod_{\mathrm{v} \in \mathcal{N}(\mathrm{c}_j) \setminus \mathrm{v}_i} \operatorname{sign}\left(\mu_{\mathrm{v} \to \mathrm{c}_j}^{(\ell)} \right) \quad (15)$$

另外，我們將各位元的交叉熵損失函數納入原本知識蒸餾架構的損失函數，因此在模型訓練期間，用於計算損失的公式為

$$L = L_{kd} + \gamma L_{ce} \quad (16)$$

在 L_{ce} 的輔助下，傳遞相對不重要甚至錯誤訊息的邊會被賦予較低的權重，等效抑制該訊息的傳遞，如此一來便可斷開可能存在的短迴圈以降低潛在的錯誤傳播風險。

藉由以上的修改，我們預期學生模型在訓練的過程中能夠一邊吸收教師模型的精華，一邊捨棄坦納圖中不重要的邊以減少短迴圈的數量，使得學生模型能夠在與 3.2 節探討的架構擁有相同深度的神經網路情況下達到更佳、甚至優於教師模型的性能。

4. 研究結果

我們將知識蒸餾法的架構套用在基於神經網路的最小和解碼器，並對其進行一系列的訓練與測試。這個專題所使用的奇偶校驗矩陣 H 是從通道編碼資料庫[8]取得的。由於這些奇偶校驗矩陣屬於右下三角矩陣，因此可以規律地透過矩陣的基本列運算(elementary row operation)化為 $H = [P^T|I_{n-k}]$ 的形式，進而得到生成矩陣 $G = [I_k|P]$ 使得 $GH^T = 0$。在模擬時，我們利用生成矩陣以及隨機產生的資訊得出一組碼字，接著將其通過可加性高斯白雜訊通道，成為解碼器的輸入。

圖 3 為 BCH(63,45)碼模擬的結果，黑色線和綠色線分別是最小和解碼器做 5 次和 30 次迭代運算(我們分別稱它們為 MS-5 和 MS-30)的模擬結果，其中 MS-30 為知識蒸餾法架構下的教師模型；藍色線對應的模型是神經補償式最小和解碼器(NOMS)做 5 次迭代運算；棕色線是運用知識蒸餾技術訓練出來的模型(KD)，我們選用的學生模型是神經補償式最小和解碼器做 5 次迭代運算；紫色線則是我們提出的修改版知識蒸餾架構(mKD)。

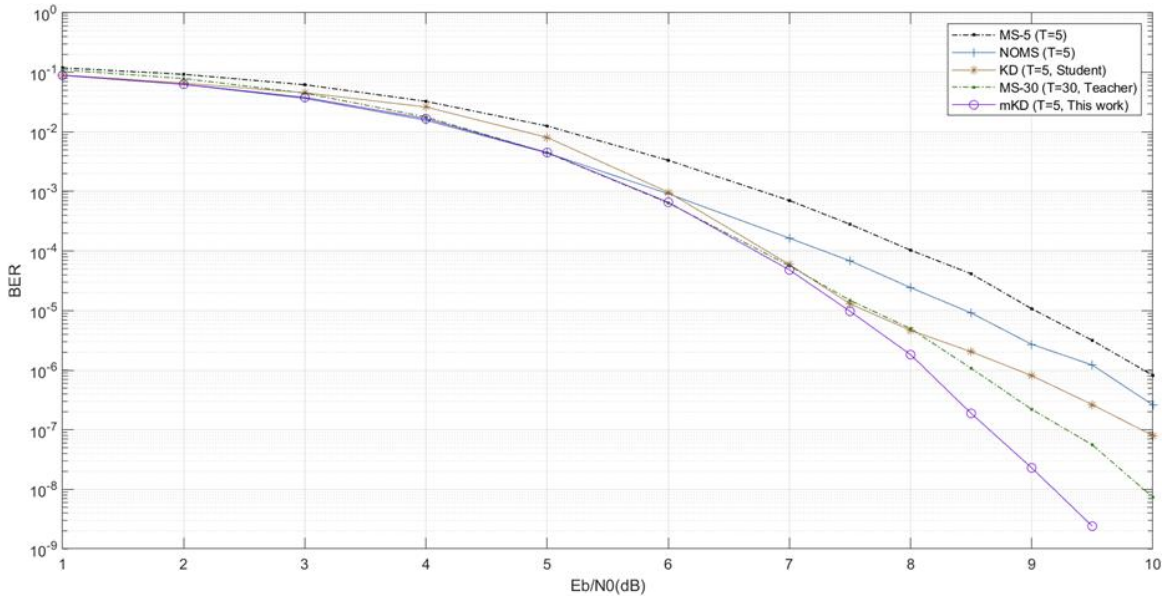


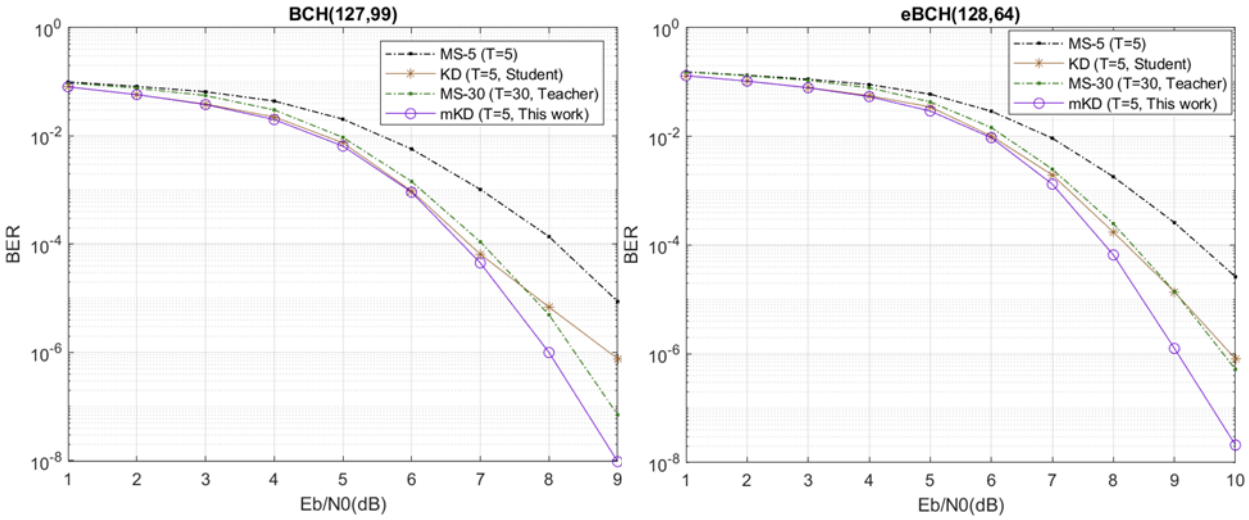
圖 3 使用 BCH(63,45)碼模擬在各種訊號雜訊比下不同模型的位元錯誤率

從上圖可看出 NOMS 在低、中訊號雜訊比的情況下$(1-6\text{dB})$，其性能相較於 MS-5 有顯著的改進，但在高訊號雜訊比 $(7-10\text{dB})$，NOMS 進步的幅度會隨著訊號雜訊比的增加而降低。KD 在中訊號雜訊比$(3-6\text{dB})$的性能雖不及 NOMS，但在訊號雜訊比為 $7-8\text{dB}$ 時，其位元錯誤率非常接近 MS-30，顯示出此時學生模型能非常成功地模仿教師模型。在更高的訊號雜訊比$(8.5-10\text{dB})$，學生模型雖然無法維持與教師模型相等性能，但仍展現優於 NOMS 解碼表現的優勢。

而我們提出的架構則同時具備 NOMS 和 KD 的優點，在低、中訊號雜訊比的情況下$(1-6\text{dB})$，mKD 的解碼能力與 MS-30 相當，這意味著 mKD 只需耗費

MS-30 六分之一的運算成本就能得到與之相同的位元錯誤率表現。在高訊號雜訊比 (8 − 10dB)，mKD 的性能甚至比 MS-30 來得好。具體來說：mKD 在訊號雜訊比為 8dB 時分別改進 MS-30、KD、NOMS、MS-5 大約 0.3dB、0.7dB、1.3dB、1.7dB 的性能，在訊號雜訊比為 8.5dB 時分別改進 MS-30、KD 大約 0.5dB、1.6dB 的解碼表現，顯示出 mKD 改進 NOMS、MS-30 的程度會隨著訊號雜訊比的增加而有進步的趨勢，這正好是 KD 缺乏的能力，也是我們認為 mKD 比 KD 模型更有價值的關鍵。

為了確認我們提出的架構具有一般性，我們也對更長的碼長如 BCH(127,99)碼，以及 5G 通訊更常使用的 eBCH(128,64)碼進行模擬，其結果如圖 4。正如我們所預期，mKD 在低、中訊號雜訊比的情況下(1 − 6dB)擁有與 MS-30 相近的性能，而在高訊號雜訊比(7 − 10dB)，mKD 的解碼表現則是明顯優於其他解碼器。



(a) (b)

圖 4 使用(a)BCH(127,99)碼、(b) eBCH(128,64)碼進行模擬以查看 mKD 是否適用於各種 BCH 碼

表 1 訓練時期所使用的超參數

超參數(hyper-parameter)	值
神經網路層數(layers)	12
優化器(optimizer)	Adam
學習率(learning rate)	0.01
批次大小(batch size)	360
訊號雜訊比範圍(SNR range in dB)	[1,10]
範數級數 p (norm order)	12
γ	10000

5. 總結

在這個專題中，我們成功地結合傳統錯誤更正碼與機器學習的創新技術設計出一個擁有高解碼性能但低運算複雜度的解碼系統。實作上，我們根據對 BCH 碼奇偶校驗矩陣特性的了解以及利用機器學習相關的基礎知識為神經置信度傳播解碼器提出了適用於各種 BCH 碼的修改版知識蒸餾法架構。透過各式的訓練與嚴謹的測試，我們提出的架構在高訊號雜訊比(7 – 10dB)能在性能上為 MS-30 帶來超過 0.5dB 的進步幅度，而且在達到相同位元錯誤率的情況下，教師模型的運算複雜度是我們提出的模型的 6 倍以上。

在此研究中，我們使用的知識蒸餾技術屬於單一教師的離線蒸餾(offline distillation)，然而在機器學習的領域中，與知識蒸餾相關的方法還有線上蒸餾(online distillation)、多教師蒸餾(multi-teacher distillation)、量化蒸餾(quantized distillation)……，這些不同的知識蒸餾方法顯示出我們提出的架構還有許多發展性，因此我們認為未來若可對修改版知識蒸餾法做更進一步的優化，則非常有機會為 5G 通訊提供另一種實現低延遲、高可靠度的可能性，同時也推動通訊系統中錯誤更正技術的進一步發展。

6. 參考文獻

[1] E. Nachmani, Y. Be'ery and D. Burshtein, "Learning to decode linear codes using deep learning", *Proc. 54th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, pp. 341-346, Sep. 2016.

[2] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[3] Q. Wang, Q. Liu, S. Wang, L. Chen, H. Fang, L. Chen, Y. Guo, and Z. Wu, ''Normalized min-sum neural network for LDPC decoding,'' *IEEE Trans. Cognit. Commun. Netw.*, vol. 9, no. 1, pp. 70–81, Feb. 2023.

[4] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun.2017, pp.1361–1365.

[5] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[6] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.

[7] E. Nachmani and Y. Be'ery, "Neural decoding with optimization of node activations," *IEEE Commun. Lett.*, vol. 26, no. 11, pp. 2527–2531, Nov. 2022.

[8] M. Helmling et al. (2019). *Database of Channel Codes and ML Simulation Results.* [Online]. Available: https://www.uni-kl.de/channel-codes

[9] H.-C. Lee, "An efficient BCH decoder for WBAN applications," *IEEE Commun. Lett.*, vol. 25, no.6, pp.1766-1770, Jun. 2021.

[10] K.Tian et al., "A scalable graph neural network decoder for short block codes," in *ICC 2023-IEEE International Conference on Communications.* IEEE, 2023, pp. 1268-1273.

[11] Jovan Milojkovic et al., "Learning to decode linear block codes using adaptive gradient descent bit-flipping," in *2023 12th International Symposium on Topics in Coding*. IEEE, 2023.

7. 心得感想

錯誤更正碼在大學部的通訊系統課程中較少被深入探討，這讓我們三位組員在專題研究初期面臨了不小的挑戰。然而，藉由從基本的通訊系統編碼模擬開始，逐步實現 LDPC 解碼器，我們最終在復現論文過程中取得了超出預期的進展，這是我們組員一開始未曾預料到的。這十個月的合作與討論中，我們在一個陌生的領域中突破自己，取得了比過去論文更好的成效，這讓我們感到非常有成就感，也獲益良多。最後，我們特別感謝指導教授翁詠祿老師及博士班學長古軒。感謝教授在每週的討論中提供的指導與建設性建議，為我們指引研究方向。也感謝學長耐心傳授錯誤更正碼的基礎知識，並在我們遇到困難時抽出寶貴時間與我們討論解決方案。