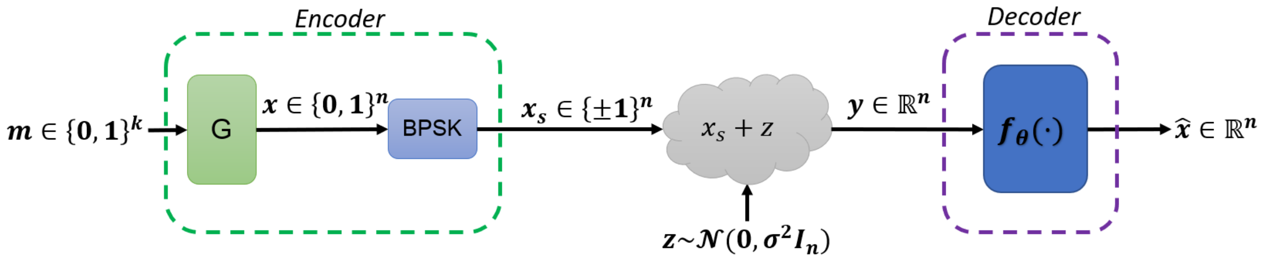
A Foundation Model for Error Correction Codes

Yoni Choukroun and Lior Wolf

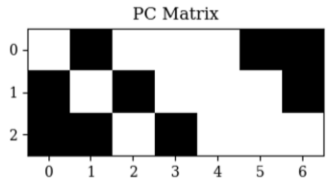
School of Computer Science, Tel Aviv University

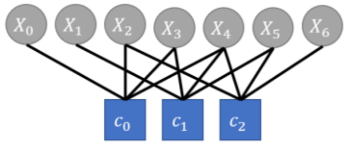
Error Correction Code: Setting

- **Goal**: *allow reliable data transmission over a noisy communication channel.*
  - A desired binary message $\boldsymbol{m}$ is encoded to a *redundant* **codeword** $\boldsymbol{x} = \mathbf{G}\boldsymbol{m}$ over ***GF(2)*** and modulated (via BPSK) to $\boldsymbol{x_s}$.
  - The noisy channel adds (AWGN) noise $\boldsymbol{z}$ such that $\boldsymbol{y} = \boldsymbol{x_s} + \boldsymbol{z}$.
  - The (parameterized) decoder $\boldsymbol{f_\theta(y)}$ *aims at retrieving the original codeword* $\boldsymbol{x}$ from $\boldsymbol{y}$.



- The **parity check** matrix $\boldsymbol{H} \in \{\mathbf{0},\mathbf{1}\}^{(n-k)\times n}$ is defined such that $\mathbf{G}^{\mathrm{T}}\boldsymbol{H} \equiv \mathbf{0} \Rightarrow \boldsymbol{Hx} = \mathbf{0}$.
  - The parity check equations allows parity check *errors discovery*: $\boldsymbol{H}(\boldsymbol{x} + \boldsymbol{z_b}) = \boldsymbol{Hx} + \boldsymbol{Hz_b} = \boldsymbol{Hz_b}$
  - The **Tanner graph** is the graph representation of $\boldsymbol{H}$ (edge for 1 in each column)

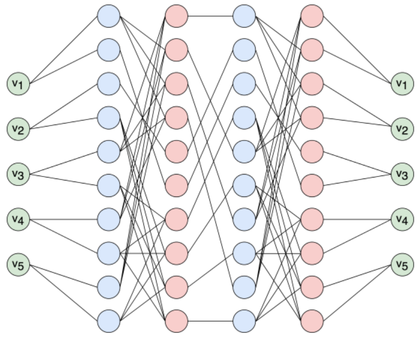




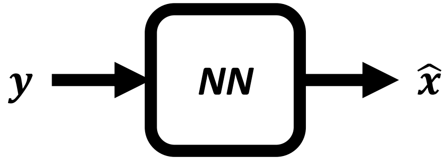
Neural Decoders

❖ ***Two** main families of neural decoders:*

- ***Model-based*** decoders implement *augmented* parameterized versions of the classical *Belief Propagation* decoder built upon the **Tanner graph** (graph representation of $\boldsymbol{H}$).
    - Pros:
        - Invariant to the codeword (robust to codewords overfitting)
        - Built on **iterative** legacy methods
    - Cons:
        - Suffers from ***heavy* and *restrictive* inductive bias**.
        - **Improvement vanishes** as the code length and the number of iterations increase



- ***Model-free*** decoders employ **general** types of neural network architectures (e.g., MLP, RNN)
    - Pros:
        - Total **freedom** in model design
    - Cons:
        1. Overfitting (exponential number of codewords for training) [1]
        2. Difficulties in learning the code [2]
        3. Lack Iterative formulation [3]



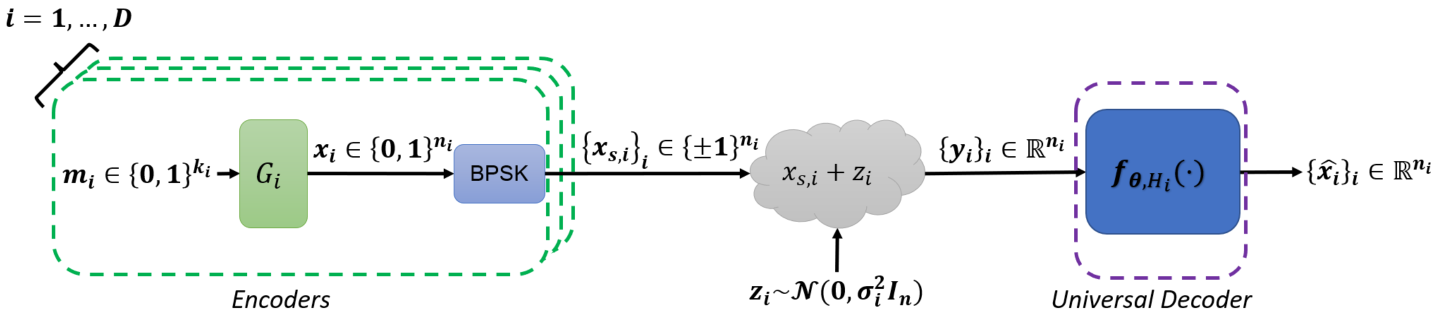
- Cons in Common :
    4. **Lack Code invariance** *(one decoder must be designed and trained for each code/rate/length)*

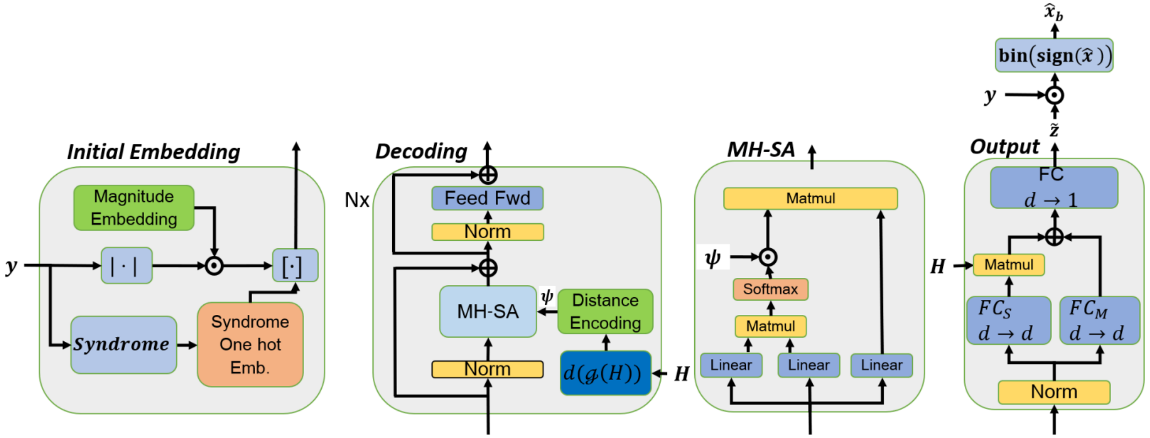
[1] *Deep learning for decoding of linear codes-a syndrome-based approach,* A. Bennatan, Y. Choukroun and P Kisilev, ISIT18
[2] *Error Correction Code Transformer,* Y. Choukroun and L. Wolf, Neurips22
[3] *Denoising Diffusion Error Correction Codes,* Y. Choukroun and L. Wolf, ICLR23

One needs to develop, train, and deploy one neural decoder for each family of code, length, and rate.

How can we develop a <u>single</u> universal neural decoder which is code/length/rate invariant?

A Foundation Model for Error Correction Codes *ICLR24*

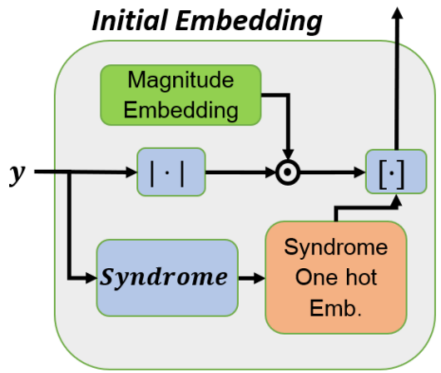




Code-Invariant Initial Embedding

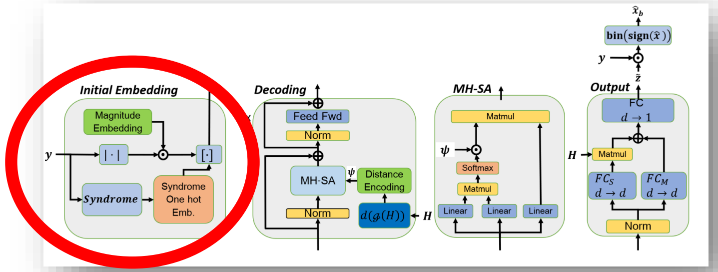
- In ECCT, a unique model is crafted for every code and length where the initial embedding is designed such that **each input bit possesses its distinct embedding vector**, providing, as a byproduct, a **learned positional encoding.**
- In our **length-invariant** model (FECCT), we propose a new code-invariant embedding, where **a single embedding** is given for all **magnitude** elements, and **two embeddings** are given for every element of the binary **syndrome**.

$$\phi_i = \begin{cases} |y_i| W_M, & \text{if } \ i \leq n \\ W^S_{(s(y))_{i-n+1}}, & \text{otherwise} \end{cases}$$



With $\{W_M, W^S_0, W^S_1\} \in \mathbb{R}^d$.

- This new **length/rate-invariant initial encoding** requires **three** embedding vectors compared to the $\boldsymbol{2n - k}$ vectors of the ECCT.
- In contrast to ECCT which captures the bit position with learned embedding, our method **lacks positional information**.

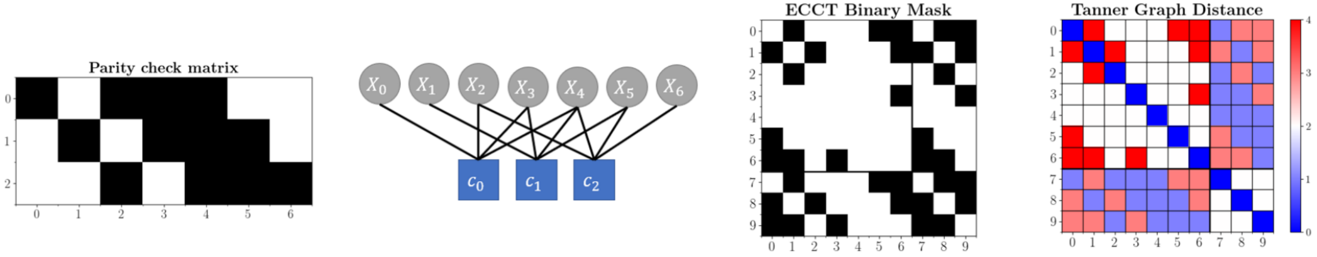


Tanner Graph Distance Masking as Code and Positional encoding

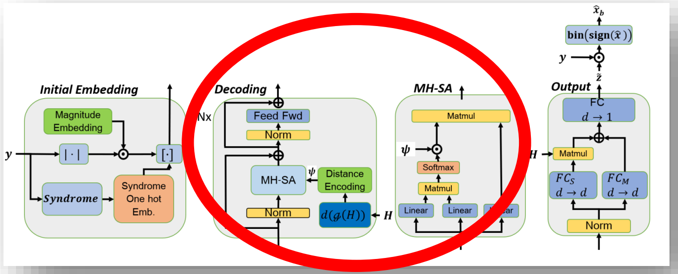
- FECCT's masking serves two purposes.
  - Similar to ECCT, it **integrates the code structure** into the transformer.
  - Adds the **relative position information** to the processed elements.
- The Tanner graph captures the **relations** between every two bits in the code (***relative** positional encoding*).
- We consider the distance matrix $\boldsymbol{\mathcal{G}(H)} \in \mathbb{N}^{(2n-k)\times(2n-k)}$, induced by the code (Tanner graph).
  - Each element (i, j) in this matrix is defined as the *length of the shortest path in the Tanner graph* between node i and node j.
- We learn a *parameterized mapping* $\boldsymbol{\psi}: \mathbb{N} \rightarrow \mathbb{R}$ of the distance matrix, incorporated into the self-attention

$$A_H(Q,K,V) = \left(\text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \odot \psi(\mathcal{G}(H))\right)V.$$

- This attention mechanism generalizes the ECCT which captured **only up to two rings** information.

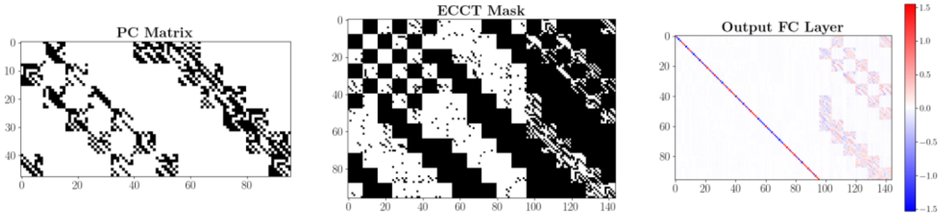


(a) (b) (c) (d)



Parity-Check Aware Prediction

- ECCT makes use of two *fully connected layers* (least length invariant modules) for the final prediction ($(2n-k) \rightarrow n$)
- ECCT's learned output layer is (surprisingly) greatly **induced** by the code/parity check matrix.

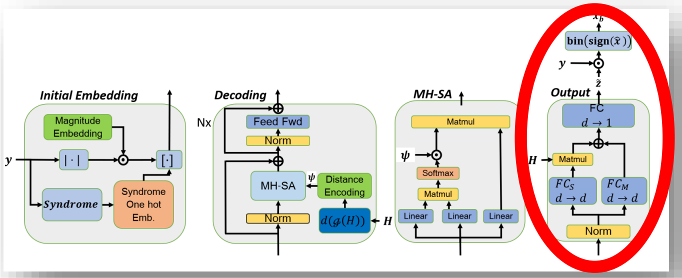


(a) (b) (c)

- Motivated by this phenomenon, we explicitly **enforce** a similar dependency structure.
- By *splitting* the syndrome and the channel output elements we integrate the remaining syndrome information by **aggregation** according to the parity check matrix connectivity

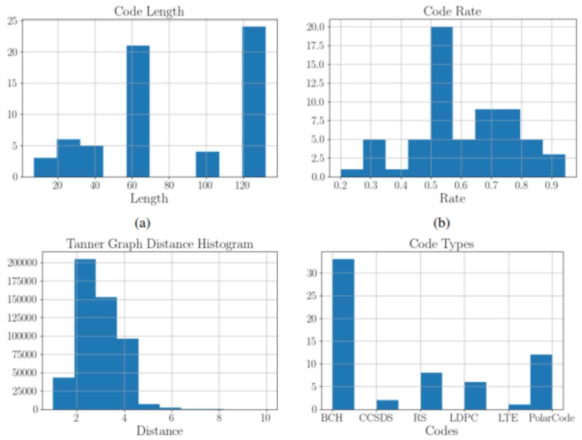
$$\hat{\tilde{\varepsilon}} = \left(\phi_{o,M} W_M + H^T(\phi_{o,S} W_S)\right) W_{d\rightarrow 1}$$

- This way, the final prediction is **code-aware** but also **code/length invariant**.
- Finally, the FECCT being invariant its *number of parameters* is **independent of the code**.



Experiments

- Trained on multiple codes, **our *single* decoder** (with smaller capacity) can match and even outperform **other methods *designed and trained separately on each code***, in multiple scenarios
  - Pretrained codes
  - Zero-shot codes
  - Fine-tuned codes

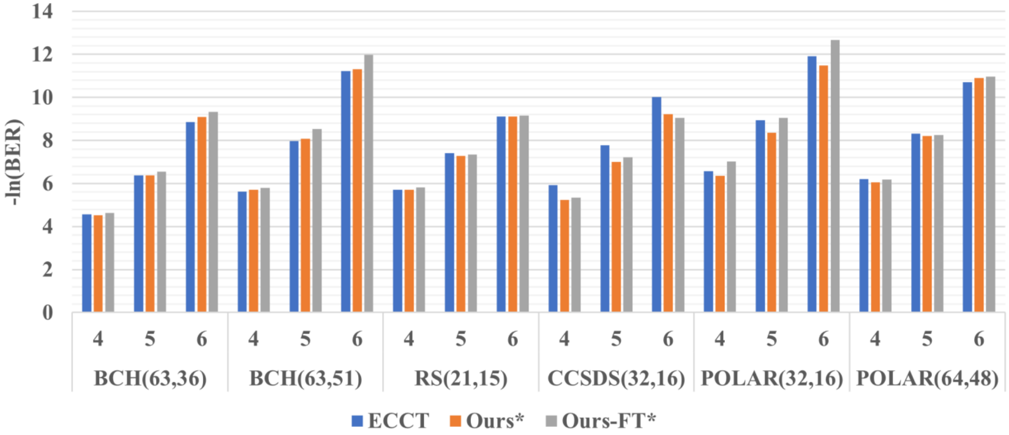


Method	BP			Hyp BP			ARBP			ECCT			Ours		
E_b/N_0	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
BCH(63,36)	3.72 4.03	4.65 5.42	5.66 7.26	3.96 4.29	5.35 5.91	7.20 8.01	4.33 4.57	5.94 6.39	8.21 8.92	4.56	6.37	8.85	4.53	6.38	9.10
BCH(127,120)	NA	NA	NA	NA	NA	NA	NA	NA	NA	4.70	6.37	8.95	4.62	6.33	8.95
Reed Solomon(21,15)	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.71	7.42	9.11	5.71	7.28	9.12
Reed Solomon(60,52)	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.53	7.54	9.98	5.47	7.59	10.21
POLAR(32,16)	NA	NA	NA	NA	NA	NA	NA	NA	NA	6.57	8.94	11.91	6.36	8.36	11.49
POLAR(64,48)	3.52 4.26	4.04 5.38	4.48 6.50	4.25 4.59	5.49 6.10	7.02 7.69	4.77 5.57	6.30 7.43	8.19 9.82	6.21	8.32	10.71	6.06	8.21	10.90

Pretrained Codes

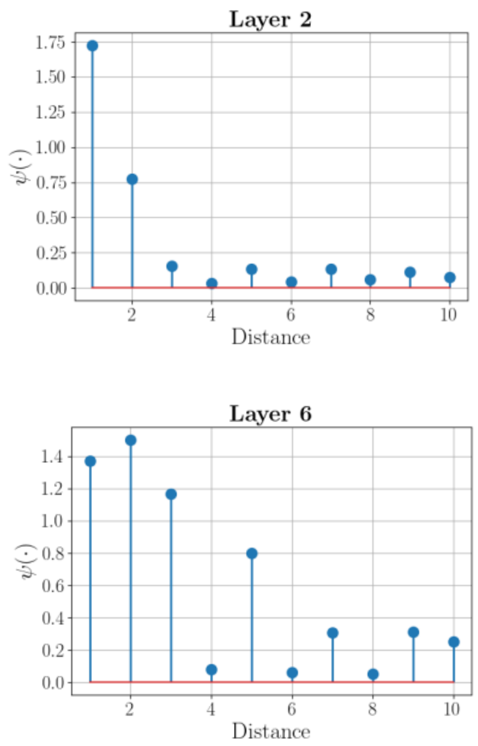
Supervision	Unlearned			Fully supervised									Zero-Shot		
Method	BP			Hyp BP			ARBP			ECCT			Ours		
E_b/N_0	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
BCH(63,45)	4.08 4.36	4.96 5.55	6.07 7.26	4.48 4.64	6.07 6.27	8.45 8.51	4.80 4.97	6.43 6.90	8.69 9.41	5.18	7.24	10.20	5.18	7.32	10.31
BCH(63,51)	4.34 4.50	5.29 5.82	6.35 7.42	4.64 4.80	6.08 6.44	8.16 8.58	4.95 5.17	6.69 7.16	9.18 9.53	5.63	7.96	11.22	5.71	8.07	11.31
BCH(127,92)	NA	NA	NA	NA	NA	NA	NA	NA	NA	4.10	5.71	8.38	4.11	5.84	8.79
BCH(255,163)	NA	NA	NA	NA	NA	NA	NA	NA	NA	3.34	4.13	5.80	3.34	4.13	5.76
CCSDS(128,64)	6.55 -	9.65 -	13.78 -	6.99 -	10.57 -	15.27 -	7.25 -	10.99 -	16.36 -	6.77	10.51	15.90	6.52	9.67	15.01
CCSDS(32,16)	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.93	7.77	10.02	5.23	7.00	9.21
POLAR(128,86)	3.80 4.49	4.19 5.65	4.62 6.97	4.57 4.95	6.18 6.84	8.27 9.28	4.81 5.39	6.57 7.37	9.04 10.13	6.39	9.08	12.70	5.53	7.90	11.29
POLAR(64,32)	3.52 4.26	4.04 5.38	4.48 6.50	4.25 4.59	5.49 6.10	7.02 7.69	4.77 5.57	6.30 7.43	8.19 9.82	6.91	9.18	12.34	5.88	7.91	10.76

Zero-Shot Codes



Fine-Tuned Codes

Analysis



Method	POLAR(64,32)			BCH(63,45)		
	4	5	6	4	5	6
ECCT	4.12	5.22	6.67	4.45	5.81	7.65
ECCT + **II**	4.27	5.54	7.14	4.52	5.98	7.92
ECCT + **IO**	4.44	5.73	7.40	4.41	5.76	7.62
ECCT + **II** + **IO**	4.09	5.26	6.80	4.31	5.62	7.41
ECCT + **DM**	4.44	5.73	7.37	4.74	6.34	8.53
ECCT + DM + **II**	4.44	5.73	7.37	5.17	7.07	9.59
ECCT + DM + **IO**	4.36	5.64	7.32	4.53	6.01	8.03
FECCT: ECCT + DM + II + IO	4.36	5.64	7.32	4.52	5.98	8.05

Method	FECCT - single			FECCT		
	4	5	6	4	5	6
POLAR(64,48)	6.35	8.50	11.12	6.06	8.21	10.96
POLAR(128,86)*	3.90	5.36	7.57	5.53	7.90	11.29
BCH(63,36)	4.01	5.42	7.30	4.53	6.38	9.10
BCH(63,51)*	4.65	6.35	8.73	5.71	8.07	11.31
Reed Solomon(21,15)	4.25	4.62	4.97	4.56	6.83	10.51
Reed Solomon(60,52)	3.68	3.81	3.77	5.47	7.49	10.24
CCSDS(128,64)*	2.90	3.42	4.30	6.52	9.67	15.01
CCSDS(32,16)*	4.10	4.54	4.43	5.23	7.00	9.21

- **Learned Distance Mapping**: FECCT seems to assign the **most** impactful mapping for the elements distanced by one and two nodes, **remarkably matching the ECCT's two-ring heuristic.**
- **Architectural Ablation** The ablation demonstrate the beneficial impact of **each of the contributions** on the obtained accuracy compared to SOTA
- **Generalization:** To show the importance of dataset diversity, we show that training FECCT on one **single** code is slightly better on the trained code but totally lacks generalization

Links

- **Papers** https://yonilc.github.io/ **:**
  - ***Error Correction Code Transformer***

    Y. Choukroun, L. Wolf – Neurips22
  - ***Denoising Diffusion Error Correction Codes***

    Y. Choukroun, L. Wolf – ICLR23
  - ***A Foundation Model for Error Correction Codes***

    Y. Choukroun, L. Wolf – ICLR24

- **Code:**
  - https://github.com/yoniLc

- **Correspondence:**
  - choukroun.yoni@gmail.com