# Slide 11

* Ban đầu định train hết 102 loài hoa nhưng vì khi thực nghiệm thì thấy **các kiến trúc model CNN đơn giản cho độ chính xác thấp nên quyết định giảm xuống 15 loài hoa**
* 15 Loài hoa này được lấy ra sau khi cho 102 loài hoa qua **model EfficienNet Transfer** , **đây là 15 loài hoa có độ nhầm lẫn lớn nhất**
* Quyết định chọn 15 loài này để xem khi đi thực nghiệm thì bọn em có thể **nghiên cứu và cải thiện được những gì** trong quá trình xây và train model

# Slide 12

* Pipe line model CNN

# Slide 13

* Hình bên là ma trận nhầm lẫn của 15 loài hoa

# Slide 14

* LeNet-5 : Model CNN đời đầu , khá là đơn giản không có gì đặt biệt
* AlexNet :

+ Model được nhóm bọn em xây dựng dựa trên các kiến trúc đã được public trên các bài báo

+ Gồm các đặt trưng

* Đầu vào là lớp 224\*224\*3
* Kế tiếp là 2 lớp Conv xen kẽ 2 lớp max pooling Conv(96,11\*11) -> Maxp() -> Conv(256,5\*5) -> Maxp() -> Conv(384,3\*3) -> Conv(384,3\*3)->Conv(256,3\*3)->Maxp()->Out(𝟔,𝟔,𝟐𝟓𝟔) (giảm W và H)
* Tiếp đến là 3 lớp Conv liên tiếp với kích thước bằng nhau là (F=384,k\*k = 3\*3)
* Rồi mới đến một lớp max pooling và sau cùng là 2 lớp dense 4096 và 1 lớp softmax 1000 nhãn (đây là kiến trúc gốc với tập dữ liệu của image net nên là softmax 1000)

# Slide 15

* GoogleNet

+ Đặt trưng bởi 2 điểm sau (Trong hình R là hàm kích hoạt – activation **Relu** , S là hàm softmax)

+ Có các khối Stem và Inception Module , với 1 Khối đầu vào là Stem và kế tiếp đó là các khối Inception module . **Trong khối Inception Module đầu vào là một tensor , tensor này cho qua các lớp Conv với các kernel có kích thước khác nhau , các kết quả đầu ra sẽ được concat lại và làm kết quả đầu ra cho toàn khối Inception .** Việc sử dụng nhiều kernel với các kích thước khác nhau (1\*1, 3\*3, 5\*5) nhằm để **bắt được các đặc trưng** ở nhiều mức độ chi tiết khác nhau

+ Đặt điểm thứ hai là khác khác với các kiến trúc CNN thông thường , Model GoogleNet có cấu trúc rẽ nhánh với mục đích là để **giúp cho Model có thể học được ở các mức độ khác nhau , tầm thấp , tầm trung và tầm cao ,** kết quả của các nhánh sau lớp softmax sẽ được concat lại và làm kết quả chính cho toàn model

# Slide 16

* Kiến trúc Model 1 . Có đặc điểm như sau :

+ Có 5 lớp Conv : 32-32-64-64-128

+ Các lớp Conv và lớp Dense đều có cùng lớp kích hoạt là Relu

+ Sau mỗi lớp Conv và Dense đều có các lớp BatchNormali

+ Dropout sau mỗi bước thay đổi quan trọng của model 32 – 64 – 128 – Dense – Softmax

# Slide 17

* Kiến trúc Model2 . Có đặc điểm như sau :

+ Có 4 lớp Conv : 16-32-64-128 , không có lớp BatchNomoli và Dropout

+ Sử dụng hàm kích hoạt Relu cho các lớp Conv và Tanh cho các lớp Dense

+ Không dùng BatchNomoli cho toàn bộ model

+ Chỉ thêm vào các lớp Dropout sau lớp Dense

# Slide 18

* Kiến trúc Model3 . Có đặc điểm như sau :

+ Có 4 lớp Conv : 16 – 32 – 84 – 128

+ Toàn bộ model sử dụng lớp kích hoạt là Relu

+ Sử dụng các lớp BatchNomoli cho các lớp Conv và Dropout cho các lớp Dense

**+ Từ model 2 qua model 3 là thay đổi bằng cách thêm các lớp BatchNomolize cho Conv và thay đổi hàm kích hoạt**

# Slide 19

* Đó là trên lý thuyết còn kết quả thực nghiệm thu được khi huấn luyện như sau

+ Với Model2 như trong hình có mức biến động ít lý do là không sử dụng các lớp BatchNomoli

Ngược lại Model3 có sự biến động mạnh **đối với tập Validation** . Với tập train thì nó vẫn có sự ổn định .

+ Model3 khác với Model2 là có BatchNomolization , cho hội tụ sớm nhưng không ổn định

+ Tuy nhiên cả 2 đều cho độ chính xác sau cùng là như nhau

+ Giải thích hiện tượng :

* **BatchNormalization thường làm cho quá trình huấn luyện trở nên ổn định hơn bằng cách chuẩn hóa đầu ra của các lớp trước đó, nhưng khi không có nó, mô hình có thể gặp ít sự điều chỉnh đột ngột về giá trị trung bình và phương sai của các lớp trung gian.**
* **BatchNormalization có thể dẫn đến sự biến động mạnh hơn trên tập validation vì nó thực hiện chuẩn hóa dựa trên thống kê từ batch hiện tại. Điều này có thể làm cho các đặc trưng đầu ra thay đổi nhiều giữa các batch khác nhau, đặc biệt khi batch size nhỏ hoặc khi có sự khác biệt lớn giữa tập train và tập validation.**

# Slide 19

* Đây là kết quả huấn luyện của model GoogleNet : Là model cho độ chính xác cao nhất .

# Slide 20

* Đọc kết quả

+ Không phải cứ lúc nào model có kiến trúc phức tạp thì mới tốt

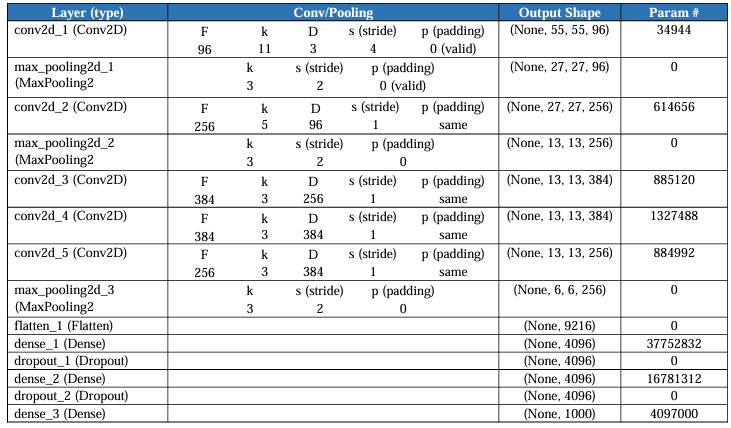
+ Trên thực tế như trong bảng thì GoogleNet cho kết quả tốt nhất nhưng nó lại có số param thấp nhất , **nó có ích trong việc tái sử dụng cho transfer learning hoặc khi xây dựng API cho ứng dụng làm giảm thời gian load model .**

# Slide 21

* Đọc kinh nghiệm rút ra

Slide 14 : AlexNet :

**Note (nằm ngoài phần thuyết trình :**



(Qua Conv thì giảm W,H và tăng D lên còn qua Maxpooling thì giảm W và H và D giữ nguyên)

(Lưu ý : Trong kiến trúc , Ta biết được kích thước các kernel của lớp Conv thông qua mỗi mỗi 11->5->3->3->3 , đây tất cả đều là kernel của Conv, s và p thì cũng không được nói rõ , cần phải dựa vào tensor input và output để tính)(Nhớ là số lượng F kernel sẽ là D của output sau khi cho qua kernel đó) (Nên ta sẽ có

(Với Maxpooling thì cũng không biết k\*k , s, p là bao nhiêu , chỉ dựa vào tensor input và output để tính ra) (Mình đã có file tính chi tiết)